AD NUMBER: AD0896846

LIMITATION CHANGES

TO:

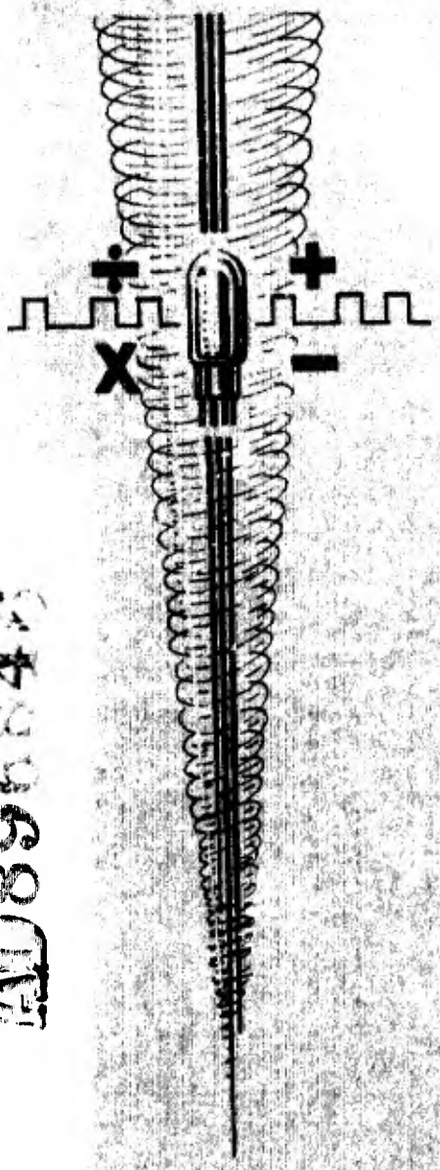
Approved for public release; distribution is unlimited.

FROM:

Distribution authorized to U.S. Gov't. agencies and their contractors; Administrative/Operational Use; 1 Nov 1947 Other requests shall be referred to the Office of Naval Research, Washington, DC 20360.

AUTHORITY

ONR ltr dtd 9 Nov 1977



PROJECT WHIRLWIND

Contract N5ori60

SUMMARY REPORT NO. 2

VOLUME 4

INTRODUCTORY MATERIAL

SERVOMECHANISMS LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

SPECIAL DEVICES CENTER

(1)

M-140

PROJECT WHIRLWIND
Summary Report No. 2.
Nov[redacted]47

(11)

(12) 170p.

(6) Project Whirlwind
Report 4.

INTRODUCTORY MATERIAL

Volume 4 of 22 volumes

(15) N5ori-76

Servomechanisms Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

220 076

Contents

INTRODUCTION

Introductory material on digital computers has been collected into this volume. The first six memorandums contain the text and illustrations of lectures given for the staff of the Electrical Engineering Department at M. I. T. The last report, R-90, discusses characteristics of the binary systems of numbers. R-115 and R-116 are papers given before winter meetings of the American Institute of Electrical Engineers and the Institute of Radio Engineers.

In considering the lecture series material, the reader is also referred to volumes 1 and 2 discussing the Whirlwind program, volumes 5, 6, and 7 on computer block diagrams and volume 8 on mathematics.

REFERENCE INDEX

M Series Memorandums

REF.	VOL.	REF.	VOL.	REF.	VOL.
M-33	8	M-95	8	M-133	18
M-46	9	M-96	9	M-134	7
M-56	9	M-99	15	M-135	7
M-60	15	M-100	8	M-136	7
M-61	8	M-101	11	M-137	7
M-62	4	M-103	16	M-138	15
M-63	4	M-105	19	M-140	4
M-64	4	M-106	11	M-141	7
M-65	14	M-107	19	M-142	8
M-66	4	M-109	16	M-143	9
M-68	15	M-110	15	M-144	10
M-69	4	M-111	7	M-145	11
M-71	8	M-112	9	M-146	12
M-72	16	M-113	7	M-147	13
M-74	14	M-114	19	M-148	14
M-76	4	M-116	16	M-149	15
M-77	15	M-117	7	M-150	16
M-78	8	M-118	16	M-151	17
M-80	16	M-119	16	M-152	18
M-81	16	M-121	9	M-153	19
M-82	16	M-123	7	M-154	20
M-83	16	M-124	8	M-155	21
M-86	14	M-127	7	M-156	22
M-89	11	M-128	16	M-157	11
M-91	15	M-129	7	M-158	7
M-92	15	M-130	9	M-159	9
M-94	8	M-131	16	M-160	8
		M-132	16	M-161	7

REFERENCE INDEX

E Series Memorandums

C Series Memorandum

REF.	VOL.	REF.	VOL.
E-7	14	E-52	19
E-24	7	E-53	13
E-31	10	E-54	19
E-32	10	E-55	19
E-33	19	E-56	15
E-37	15	E-57	15
E-38	19	E-58	19
E-39	15	E-59	19
E-41	15	E-60	19
E-42	15	E-61	16
E-44	19	E-63	19
E-45	19	E-64	15
E-47	15	E-68	13
E-48	19	E-69	15
E-49	19	E-71	19
E-50	16	E-73	16

REFERENCE INDEX

R Series Memorandums

REF.	VOL.	REF.	VOL.
R-36	14	R-115	4
R-49	14	R-116	4
R-63	14	R-117	16
R-64	3	R-118	16
R-89	19	R-120	10
R-90	4	R-121	19
R-94	14	R-122	18
R-98	14	R-123	17
R-100	14	R-124	11
R-103	14	R-125	14
R-104	16	R-126	19
R-106	15	R-127	5
R-108	15	R-127	6
R-109	19	R-128	10
R-110	9	R-129	12
R-111	15	R-130	9
R-113	15	R-131	10
R-114	8	R-132	10

MEMORANDUM NO. M-62-1

5345
Page 1 of 9

WRITTEN BY: Jay W. Forrester

SUBJECT: The Project Whirlwind Program of Electronic Digital Computation

DATE: March 12, 1947

Illustrations:
A-30337 A-30350
A-30338 A-30351
A-30353 A-30345
A-30340 A-30352
A-30341 A-30339
A-30342 A-30354
A-30343 A-30355
A-30344 A-30356
A-30346 A-30357
A-30347 A-30358
A-30348 A-30359
A-30349

This first discussion is to introduce the nature of electronic digital computation and to establish the vocabulary and basic concepts necessary for the following series of lectures.

In the last several electrical engineering seminars, we have heard discussions of the differential analyzer and analog computers. First then, let us distinguish clearly between analog and digital type computing equipment. The analog computer represents numerical values by the magnitude of a physical measurement, as for example, the voltage in an A-C network analyzer or the magnitude of a shaft rotation in the differential analyzer. A digital computer, on the other hand, uses discrete numerical values and arithmetic type calculations. In illustration A-30337 we see a collection of analog type computing elements. A separate circuit component is used for each computing operation, and we note in the illustration available equipment for the processes of addition, amplification, integration, introduction of time delay and inversion of signal. In illustration A-30333 is a typical hookup of such computing elements for the solution of a problem in ordinary differential equations. The nature of the problem is described by the physical circuit connections of the system. As herein illustrated numerical values are represented by voltages on the interconnecting lines. The solution of a differential equation by this method is theoretically exact because the network is an exact and continuous representation of the problem to be solved. However, practical and physical limitations are introduced by electrical noise level, mechanical backlash, circuit non-linearities, etc. As a result, precision of one part in one thousand is good for analog equipment and one part in ten thousand is perhaps the best obtainable.

Referring now to illustration A-30353, we see the block diagram of a digital computer. The physical connections of such a system are permanent and not altered for the problem to be solved. Quantities are handled in the computer as numerical values and can be carried to any required number of decimal places. A digital computer requires the five principal functions illustrated. The computer solves mathematical, scientific, and engineering problems by carrying out a series of arithmetic steps, one at a time. The arithmetic element can carry out the basic operations of arithmetic and other essential functions to be described later. Storage provides the required "memory" for retaining the program which controls the series of numerical steps to be executed and likewise provides a means for storing partial numerical results during the process of a computation.

The control element provides the timing signals for the remainder of the computer. In operation it extracts a control order from storage and in response to the coded information therein accomplishes the next required numerical step in the computation.

The input mechanism provides a communication means between the human operator and the computer. Problems in the form of initial data and controlling program will be prepared in coded form external to the computer on photographic film or wire tape and read into the storage element of the computer by the input mechanism.

The output from the computer may be in several forms: typewritten for isolated results, graphical for engineering data lending itself to this method, and digital or numerical as coded information on photographic film. This numerical information on film can later be used by the computer itself through the input device or can be decoded and typed by a transcribing unit.

It is readily recognized from the preceding discussion that a digital computer is able to do only computing operations that are possible for a human operator and desk calculating machine. In illustration A-30340 is shown the correspondence between the automatic digital computer and a human computer. The input provides original data and instructions for the problem to be solved. This information is stored in a notebook corresponding to the computing machine storage, and in the notebook are likewise entered partial numerical results arising during the operation. The computing machine corresponds to the arithmetic element carrying out additions, multiplications, divisions and other basic processes. The operator acts as the controlling element to read instructions from the notebook storage and to carry out the transfer of numerical values between storage and the arithmetic element. Delivery of final results corresponds to the machine output.

One of the oldest forms of digital computing equipment is illustrated in A-30341. In 1823 the British Government first financed the construction of a large scale automatic digital computing machine based on mechanical operation. This machine was never completed because of inadequate machine tools and production facilities at that time. Digital computing equipment progressed through the adding machine, various forms of desk calculators, and punched card business machines to the Harvard Automatic Sequence Control Calculator. Mechanical equipment is inherently slow and digital computing equipment has been built using relays, and still faster and more flexible equipment is being designed with electronic circuits.

Returning now to further comparisons between analog and digital equipment, we note in A-30342 the usual differential analyzer schematic for solution of a second order linear differential equation. An electrical simulator circuit for the same problem is illustrated in A-30343. These analog methods of solution employ a continuous flow of data and continuous change of variables.

In A-30344 are shown the steps of a numerical solution for the second order differential equation with zero damping. It is assumed that all variables are known at time, t_n and the previous time intervals t_{n-1}, t_{n-2}, etc. A curve for y is first extrapolated to time t_{n+1} using the first formula. This is the formula for second order extrapolation using the value of y at t_n and t_{n-1} and the value of $\dot{y}$ at time t_n. The second formula is then used to obtain the upper shaded area giving the increment in the curve $\dot{y}$. A second numerical integration for the lower shaded area gives the increment in curve $\ddot{y}$. The negative of this last point is used as a correction for the original extrapolated value of y_{n+1}. Dr. Loud, in his lecture of this series, will discuss numerical integration and its convergence considerations in more detail. Depending on the nature of the problem, iterative procedures of integration and more elegant extrapolation and integration formulas may be used.

As an illustration of digital computing machine operation, let us consider the solution of the second order determinant in A-30346. Before setting up the solution of this problem, refer to A-30347 for greater detail in the digital computing machine block diagram. Here we illustrate the arithmetic element as set up for multiplication using the A register, AR, for the multiplicand, the B register, BR for the multiplier and the accumulator register, AC for the product. Storage is likewise shown divided into many separate registers numbered for identification purposes in numerical order.

In A-30348 is shown the required program for solving the second order determinant problem. The information appearing in storage registers S1 through S11 is set up ahead of time on film and read by the input mechanism of the machine into the corresponding storage locations. In the storage registers S1 through S4, are located the numerical values of a, b, c, and d representing the initial data of the problem. Storage registers S5 through S11 are used for the required control orders. It should be borne in mind that the information in a storage register may, depending on how it is used, represent either a numerical value or a control order coded in numerical terms. The steps in the control program are itemized below and are carried out in the order in which they appear in storage.

S5. Transfer contents of register S2 to multiplier register BR. This means that the value (b), stored in S2, is transferred to the multiplier register of the arithmetic element.

S6. Transfer contents of S3 to the register AR and multiply. This means that the numerical value (c) is transferred to the multiplicand register of the arithmetic element and is multiplied by the number previously transferred to the multiplier. At the end of this operation, the product resides in the accumulator register.

S7. Transfer number in accumulator to storage S12. This means that the product (bc) which was in the accumulator is transferred to an empty storage register S12. We must next form the product (ad).

S8. Transfer contents of S1 to multiplier register BR. This means that the numerical value of (a) is moved from storage location S1 to the multiplier register of the arithmetic element.

S9. Transfer contents of S4 to register AR and multiply. This means that the numerical value (d) is transferred to the multiplicand register of the arithmetic element and multiplied by (a) which has been left in the multiplier register. At the end of this operation, the product (ad) is in the accumulator register AC.

The next operation will be the indicated subtraction of (ad-bc). The product (ad) is in the accumulator register AC. (ad) can remain in the accumulator register and the next operation will move the previously formed product (bc) from its location in S12 to the register AR

S10. Transfer contents of S12 to the register AR and subtract. At the end of this operation the difference (ad-bc) is in the accumulator register.

S11. Transfer the contents of accumulator to the output. This means that the final result of the computation is available at the machine output.

It will be observed that the use of storage register S12 for the partial result (bc) was not necessary since the numerical value of (b) in storage register S2 had already been used and the newly formed product might have been stored in S2.

Such a problem as the previous example is obviously too simple to justify use of a large scale digital computer. However, many problems of scientific and engineering importance involve computations so extensive that the facilities of illustration A-30349 become inadequate, too expansive, too costly, and consume so much time that the results when available are no longer of interest.

Problems too extensive for manual calculating machine solution are represented by the ordinary differential equations describing servomechanisms and automatic control systems including non-linear characteristics of circuit elements, backlash, and Coulomb friction; by the partial differential equations of electromagnetic fields and heat transfer; by the algebraic equations with complex coefficients representing alternating current networks and stress and vibration problems; and by the multiplication of matrices and summation of series involved in other problems.

The basic operations to be performed by a digital computer are tabulated in A-30350. The computer solves problems on an arithmetical basis using the processes of addition, subtraction, multiplication, and division. Other operations such as integration and extraction of roots can be performed by series and iterative procedures as will be described in the next lecture by Mr. Everett. Beyond the basic arithmetic operations two other functions, comparison and the substitution order, are essential to the efficient and flexible use of a digital computer. The comparison order makes possible a choice between two alternate computing sequences depending on the outcome of the numerical results of the computation itself. The substitution order makes possible the insertion of computed results into the controlling program for the computation. Necessity for the comparison order can be illustrated in the handling of Coulomb friction and necessity for the substitution order can be illustrated in the process of interpolation.

Consider first the Coulomb friction problem illustrated in A-30551 where the friction force is equal to or less than a specified maximum force. Acceleration of the mass results from the summation of Coulomb and externally supplied forces. Steps in the solution of this

problem using comparison orders is illustrated in A-30346. For the purposes of this discussion, assume that the computer can distinguish between the two cases:

1. the number is greater than zero
2. the number is equal to or less than zero.

Determine in the first comparison of the illustration if the velocity is positive. If so, the friction force equals $-F_{max}$ and we proceed to evaluate the equation. If on the other hand velocity is equal to or less than zero, we must distinguish between these two possibilities. First add the smallest possible increment to the velocity in order that the zero velocity case will become equal to +1. If V_a is now less than or equal to zero we know that V is less than zero and the Coulomb friction force equals $+F_{max}$. The alternate possibility in the second comparison is that V_a is greater than zero indicating that V equals zero and we must now evaluate the relationship between the applied force and the maximum possible friction force. In the third comparison, if F is greater than zero, we then form the quantity $(F-F_{max})$. If this quantity is positive, the mass is being accelerated. If this quantity is zero or negative, the force F is unable to overcome the Coulomb friction force F. A final similar comparison is carried out if the applied force F is zero or negative. The computing machine would be so arranged that the sequence of operations shown on A-30346 would be terminated as soon as a successful comparison was reached.

The interpolation example shown in A-30352 illustrates the need for a substitution order. The numerical values of uniformly spaced points along the arbitrary function f(x) can be stored in the first available series of storage registers as for example S27 through S32. Assume we wish to use linear interpolation

$$f(x) = f(x_N) + n\left[f(x_{N+1}) - f(x_N)\right]$$

To obtain a value of the function f(x) at any value of x, we must use a substitution order to extract the proper two numerical values from storage for use in the linear interpolation formula. Take for example x=2.45. This information on the value of x was unavailable when the problem was set up so that the controlling orders must of necessity have been left blank. We will first separate the whole digits from the fractional part of x. Store n=0.45 in register S53. Add 27, which is the identification of the first storage register devoted to the desired function, to N. This sum (27+N) represents the storage register in which will be found the desired numerical value of $f(x_2)$. This sum is

inserted into the blanks of the orders in S42 and S44. The sum ($x_2 - x_1$) is inserted in the order found in storage register S41. Having filled the blanks in orders S41, S43, and S44, we will now note that the series of operations is complete and specific. S41 transfers the numerical value of $f(x_3)$ from S30 to the accumulator, AC. Order S42 transfers the numerical value of $f(x_2)$ from S29 to the register, AR, and subtracts leaving in the accumulator the difference $\left[f(x_3) - f(x_2)\right]$. Order S43 transfers the incremental value, n, from S53 to the A register and multiplies by the value in the accumulator to obtain the result Δf. Order S44 adds the value of $f(x_2)$ transferred from storage register 29 to Δf which is in the accumulator AC. In the accumulator there then results a numerical value of f(x) according to the above linear formula at x=2.45.

Returning now to the digital computer block diagram we observe a slightly expanded version in A-30339. Here control lines have been added between the control element and the other blocks of the computer. Video electronic pulses are transmitted over these control lines and the digit transfer bus to accomplish the arithmetic operations previously outlined. Numerical values will be handled in the WWI and WWII computers in the binary system of numerical notation.

Illustration A-30354 shows the correspondence between the decimal and binary systems. It is to be noted that only two binary digits exist: 0 and 1. The binary system is then well adapted to electronic circuits in which a zero can be represented by absence of a video pulse and 1 can be represented by the presence of a voltage pulse or by the conducting state of a vacuum tube. Successive columns of binary places represent increasing powers of the base 2 just as successive decimal columns represent increasing powers of base 10.

In A-30355 we have the significant characteristics of the binary system. A number represents digits in powers of the base 2. It has a simple multiplication and addition table suitable for electronic circuit computation. A number represented in the binary system will require approximately 3-1/3 times as many digits as the same number in the decimal system. Only the digits 1 and 0 are required.

Two methods of transmitting binary digits electrically are illustrated in A-30356. In the upper illustration the presence or absence of digits in time sequence represent a binary number. By observation of the pulse arrival time, any number can be transmitted over a single conductor. In the lower illustration, parallel digit transmission of the same number is shown. In this example digits are transmitted at the same

time instant over separate cables for each digit of the number. This parallel digit transmission system will be used in WWI and WWII computers because of the reduction in number transmission time.

Illustration A-30357 shows how a twin grid vacuum tube can be used as a gate circuit or as a circuit for producing the results of the multiplication table in A-30355. Mr. David Brown will discuss circuits in some detail in his lecture.

Storage of controlling orders and numerical quantities is, of course, one of the most important functions of the digital computer. Storage for the Whirlwind I and Whirlwind II computers is being planned in the form of electrostatic storage. A sketch of this storage system is shown in A-30358. By proper control of electrode voltages and beam current, it is possible to store electrostatic charges on the dielectric surface and to later read the polarity of these charges in the output circuit. Storage for satisfactory periods of time has been observed, and good output signal level has been obtained. Many research problems principally associated with the control of secondary electron redistribution still remain to be solved.

The size or rating of a large scale digital computer can be described by two quantities - the computing speed, and the digit storage capacity. The computing speed will determine how long the solution of a particular problem will require, and the storage capacity will determine the magnitude of problem which can be readily undertaken. Two kinds of storage capacity will be used in most digital computers. The internal high-speed storage will provide high-speed access to those orders and numerical quantities required for immediate use. A slower more extensive external storage in the form of punch tape, magnetic wire, photographic film, or phospher film will be used for information not required for relatively longer times. Illustration A-30359 illustrates some of the more important comparisons between existing and proposed digital computing machines. The first three machines are completed and in operation. The fourth is in the final stages of assembly. The last five are being planned at the present time. The form and approximate capacity of internal storage is given based on the best available information. Some of the systems use a fixed decimal point in numerical notation, others a floating decimal point. Some use the decimal system, others the binary as well as other variations. These different types of number storage systems have been converted to equivalent binary capacity for comparative purposes. Multiplication speeds are approximate based on the latest available information for each of the computers. Circuits of the Whirlwind I computer will use the same computing impulse rate as planned for the Whirlwind II. Since, however, the Whirlwind I computer will use a number length of sixteen binary digits which is inadequate for

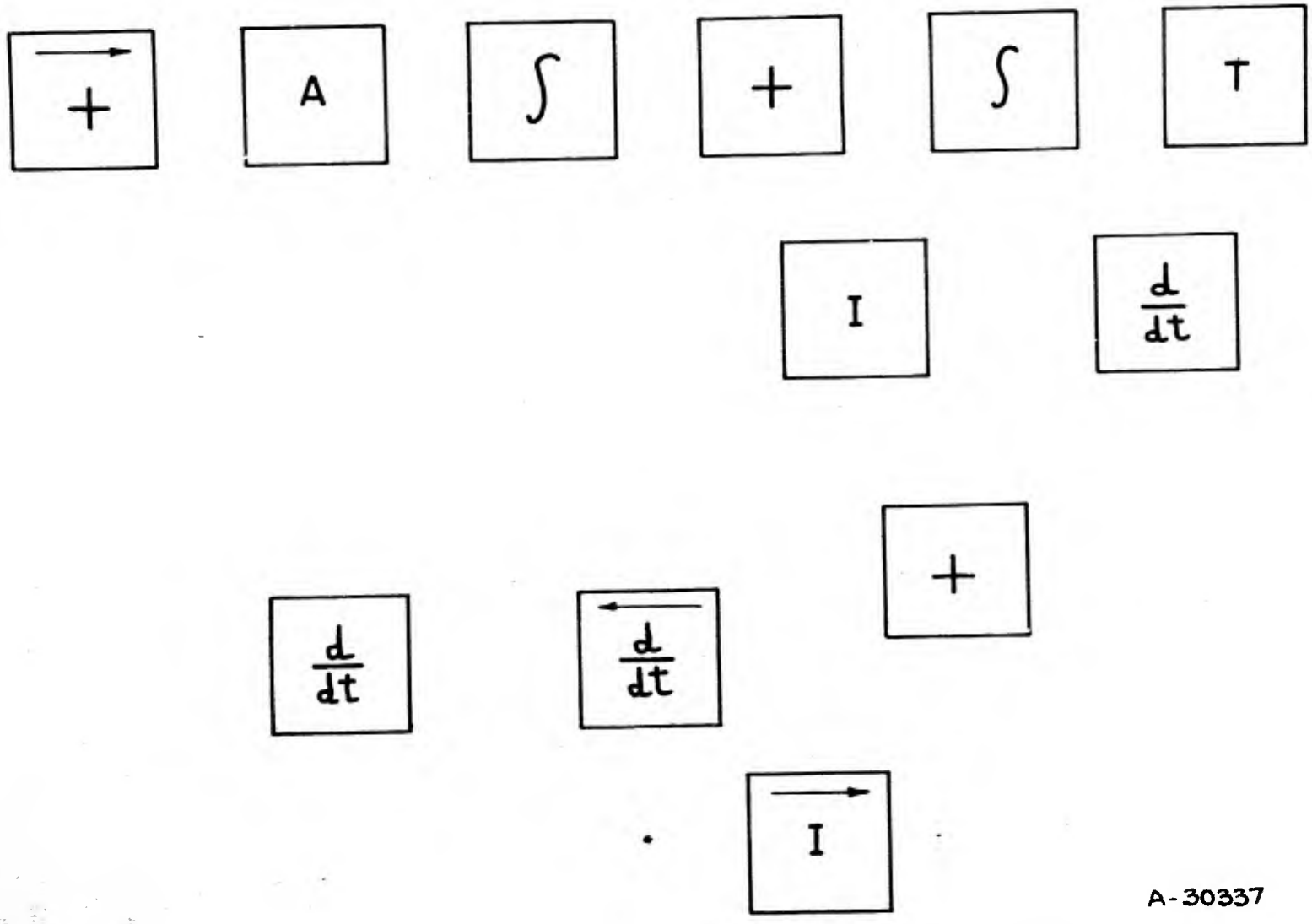
most mathematical work, double length multiplication in the form (a+b)(c+d) will be required and four multiplications must be performed. This compound multiplication will be carried out with no more program orders than required in the Whirlwind II computer, but more time will be required for the execution of a complete multiplication.

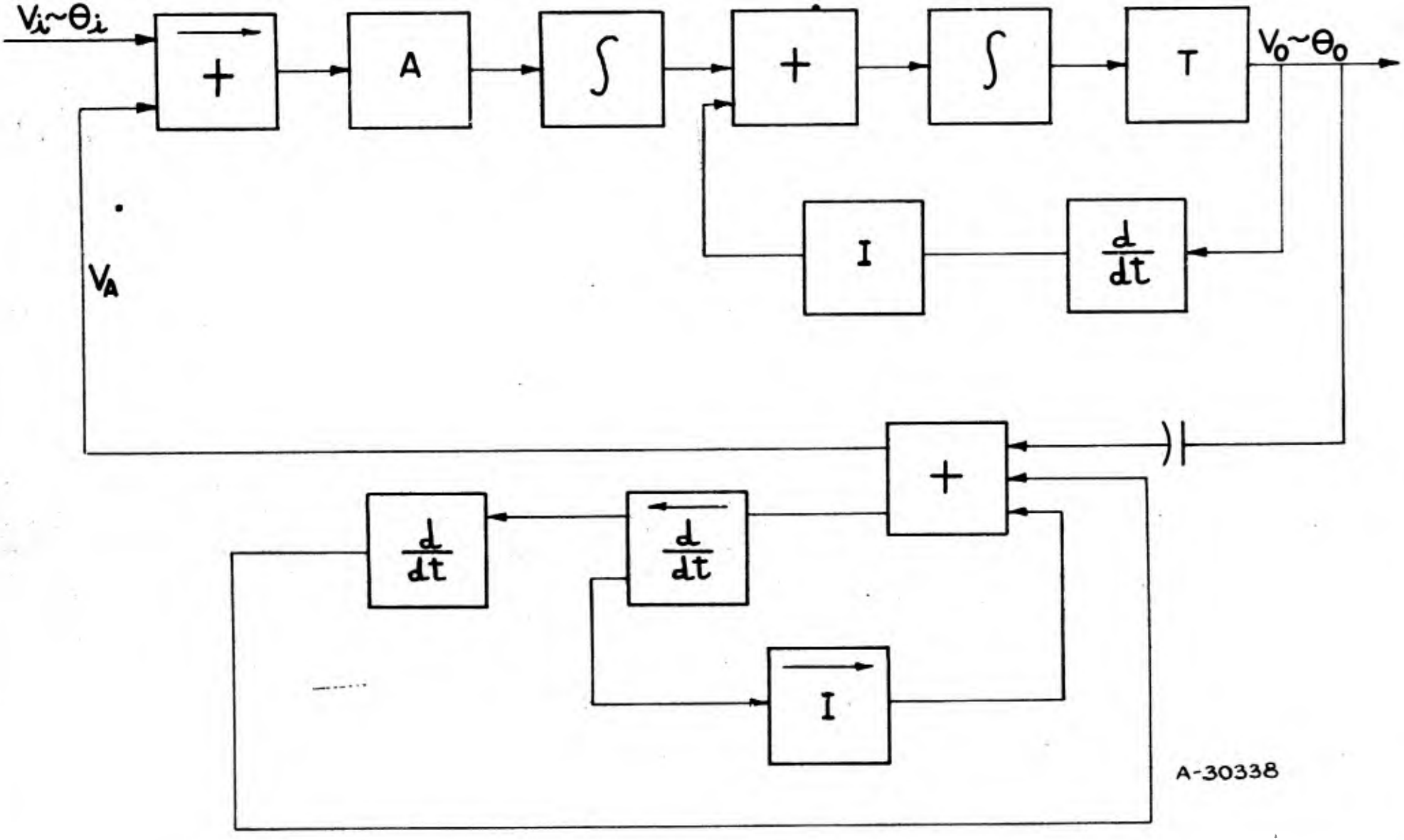
The Whirlwind I computer is being designed for study and demonstration of computing circuits and trouble shooting methods, and the computer will be used for mathematical research and study of engineering problems. Design of the Whirlwind I computer is now in progress and construction will begin this year. Laboratory testing is expected to continue through 1948, and the equipment should be ready for mathematical research work in 1949. Design of Whirlwind II which is the final large scale objective of this project will start as soon as possible, probably in 1948 and will require relatively longer to construct and test than will the Whirlwind I.

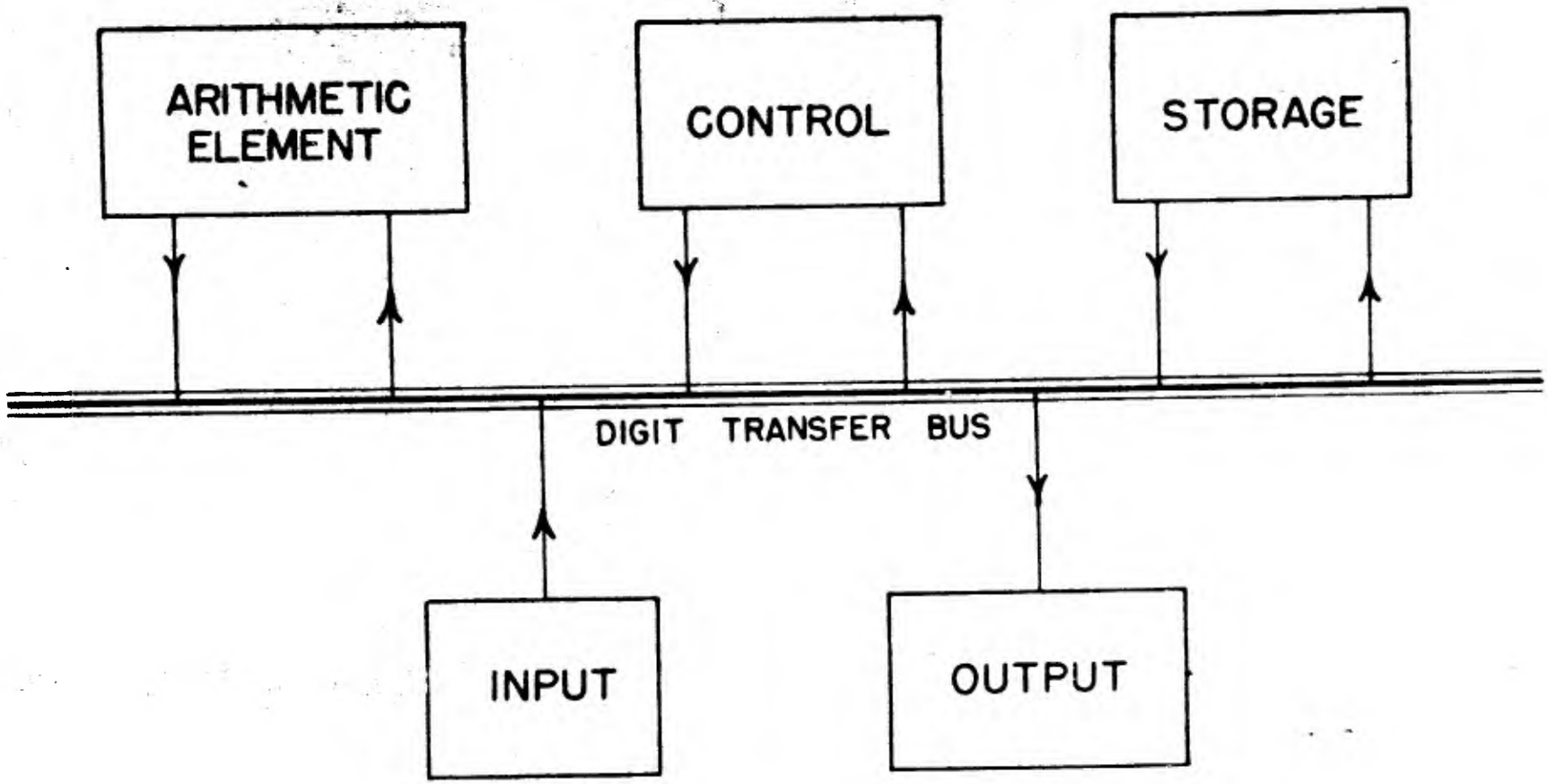
Much mathematical research must be completed before a suitable background of information is available to make proper use of a computer with the Whirlwind II capacity. It is hoped that the foundation for this research can be laid immediately and that Whirlwind I can make valuable contribution to the knowledge of high-speed numerical analysis and digital computation.

Jay W. Forrester

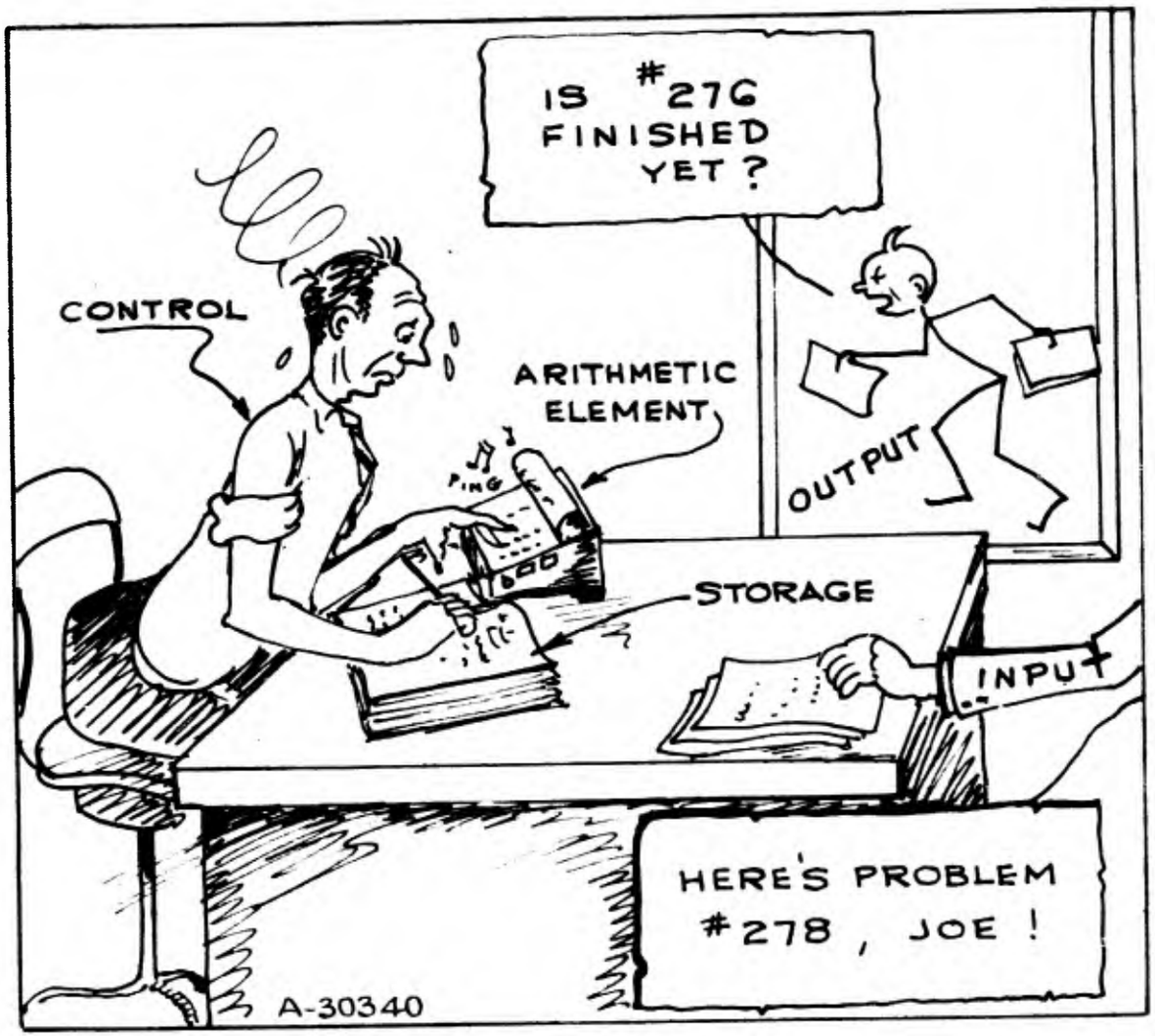
JWF:vh






ARITHMETIC ELEMENT
CONTROL
STORAGE
DIGIT TRANSFER BUS
INPUT
OUTPUT

6345 A 30353

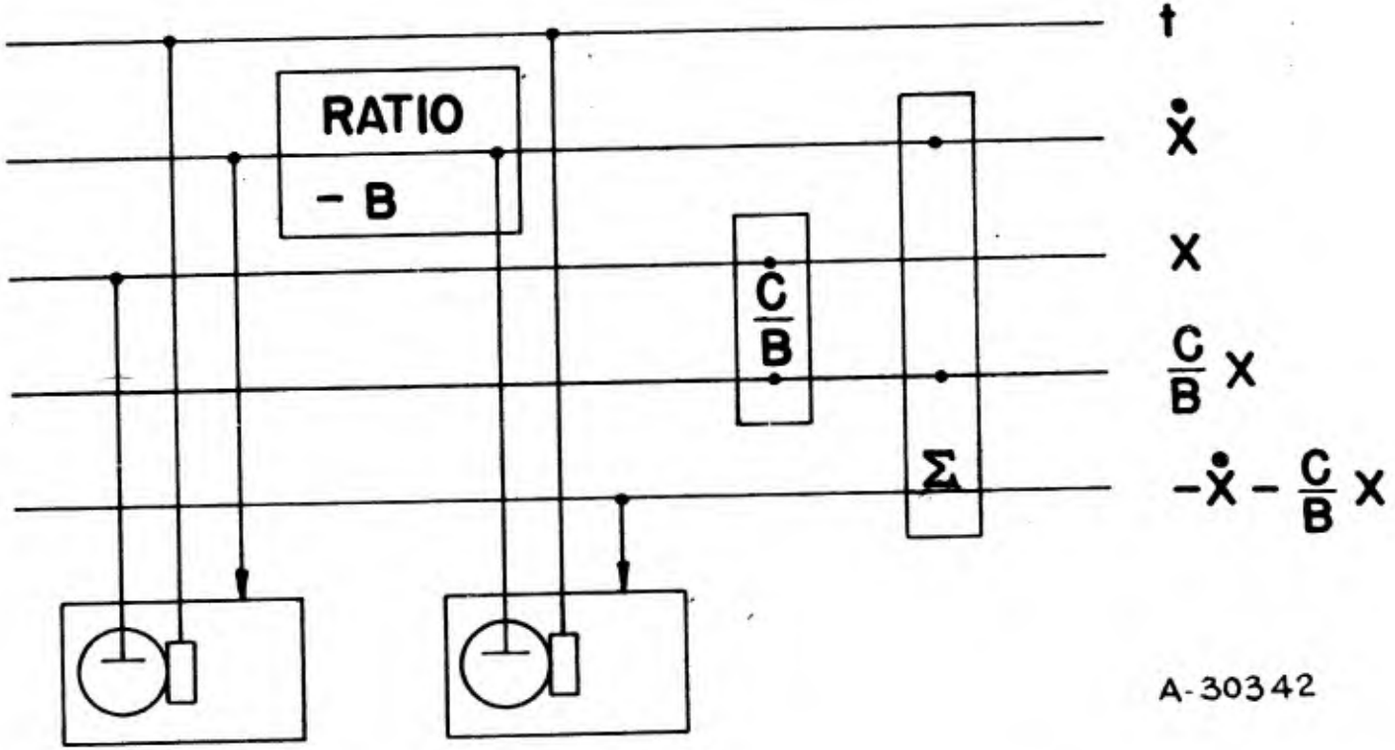

IS #276 FINISHED YET?
CONTROL
ARITHMETIC ELEMENT
PING
OUTPUT
STORAGE
INPUT
HERE'S PROBLEM #278, JOE!
A-30340

6345 A-30340

A. 30341

DIFFERENTIAL ANALYZER SCHEMATIC

$$\frac{d^2x}{dt^2} + B\frac{dx}{dt} + Cx = 0$$

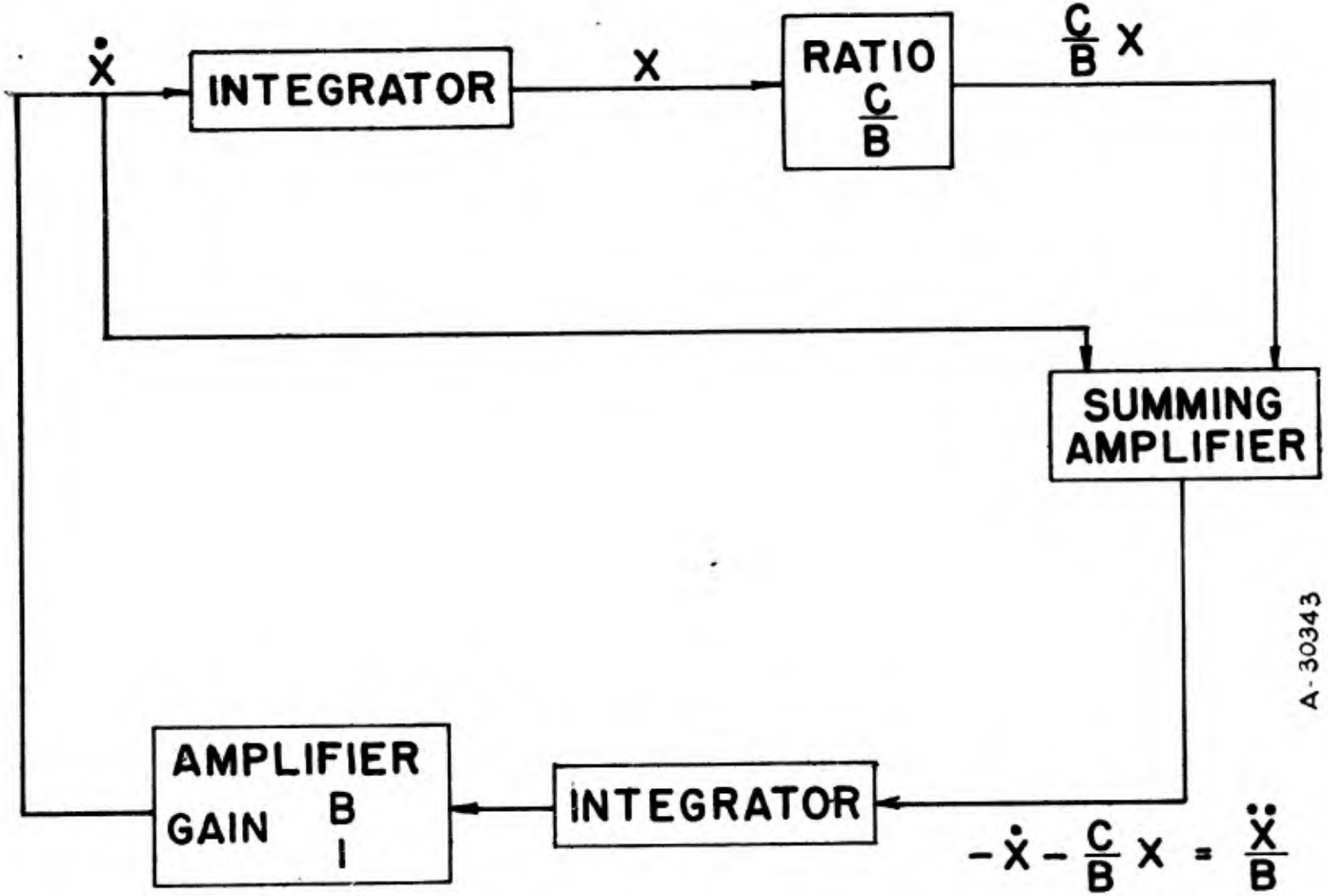


A-30342

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
SERVOMECHANISMS LABORATORY
D.I.C. NO. 6345
DR.
A-30342

SIMULATOR SOLUTION OF QUADRATIC

$$\frac{d^2x}{dt^2} + B\frac{dx}{dt} + CX = 0$$



NUMERICAL SOLUTION

$$\ddot{y} = -y$$

KNOWN: $\ddot{y}, \dot{y}, y$, at t_n and t_{n-1}

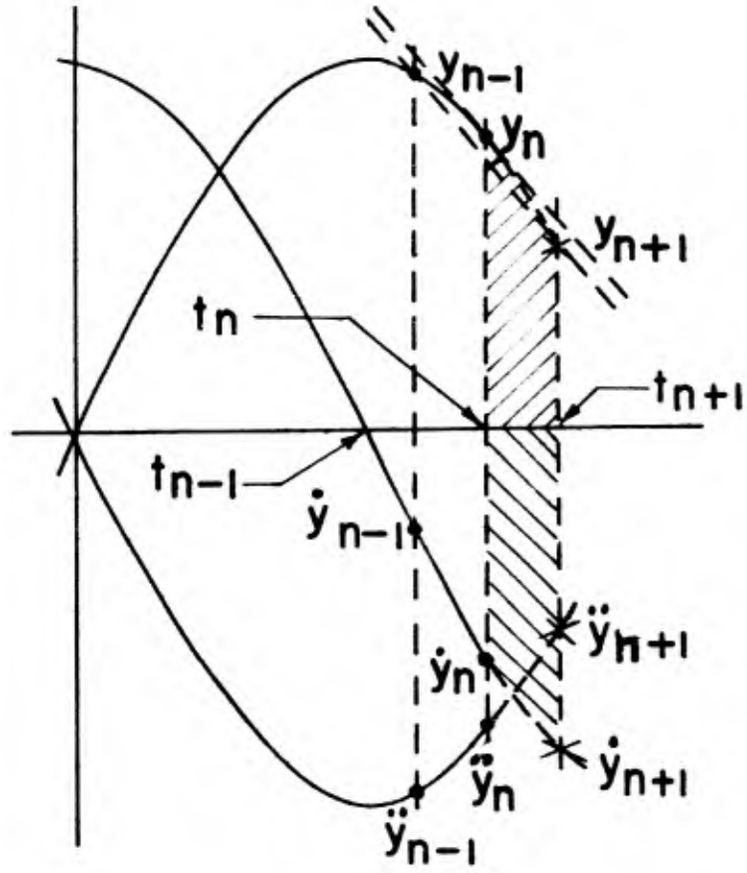
$$y_{n+1} = y_{n-1} + 2h\dot{y}_n$$

$$\Delta \dot{y} = -\frac{5y_{n+1} + 8y_n - y_{n-1}}{12} h$$

$$\Delta \ddot{y} = -\frac{5\dot{y}_{n+1} + 8\dot{y}_n - \dot{y}_{n-1}}{12} h$$

$$y_{n+1} = -\ddot{y}_{n+1}$$

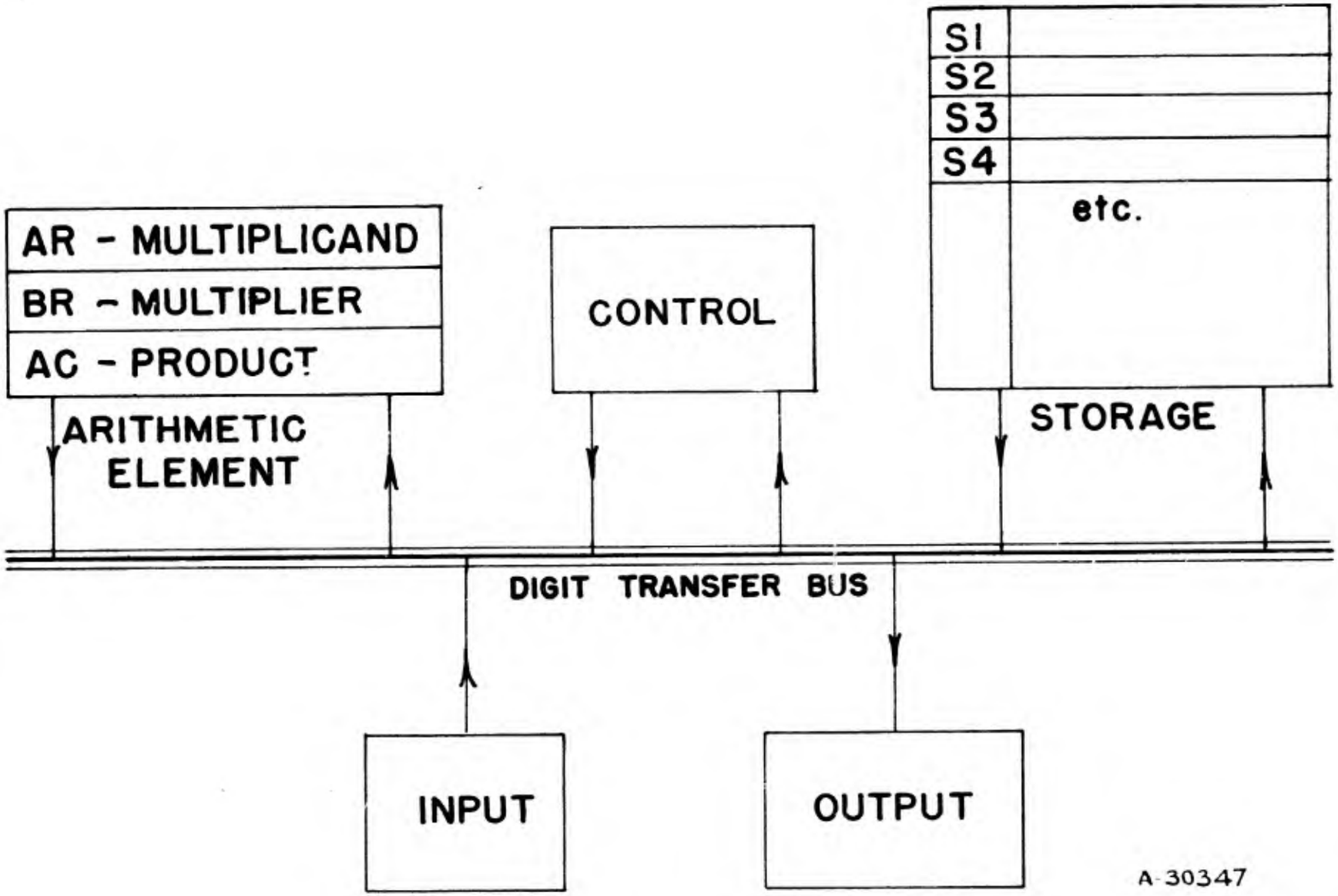
A-30344



A-30346

SOLUTION OF DETERMINANT

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - cb$$



A-30347

	REGISTER	INFORMATION IN REGISTER
INITIAL DATA	S1	a
	S2	b
	S3	c
	S4	d

PROBLEM

$$\begin{vmatrix} a & b \\ c & d \end{vmatrix} = ad - bc$$

PROGRAM

S5	t	S2	to	BR
S6	t	S3	to	AR AND MULTIPLY
S7	t	AC	to	S12
S8	t	S1	to	BR
S9	t	S4	to	AR AND MULTIPLY
S10	t	S12	to	AR AND SUBTRACT
S11	t	AC	to	OUTPUT

PARTIAL RESULTS

S12 (USED FOR PARTIAL RESULTS)

A-30348

A-30349

BASIC COMPUTER OPERATIONS

1. ADD

2. SUBTRACT

3. MULTIPLY

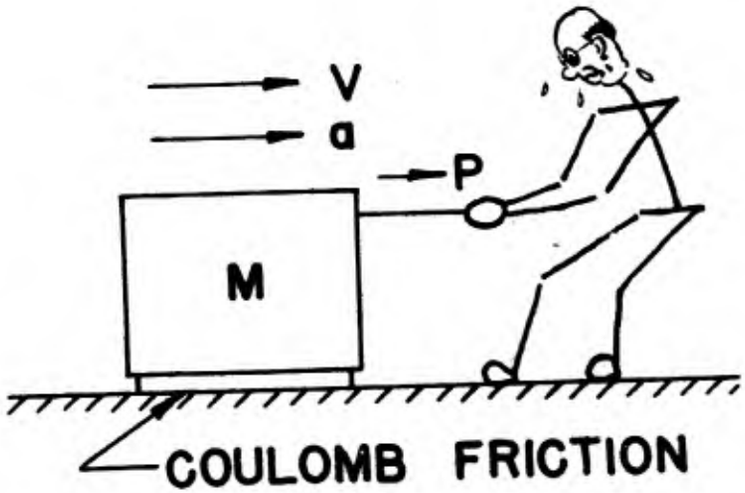
4. DIVIDE

5. COMPARE — CHOICE

6. SUBSTITUTION ORDER

A-30350

COMPARISON OR CHOICE
COULOMB FRICTION



$$|F| \leq F_{max}$$

$$M\frac{dx^2}{dt^2} = P(t) + F(t) = f(t)$$

A-30351

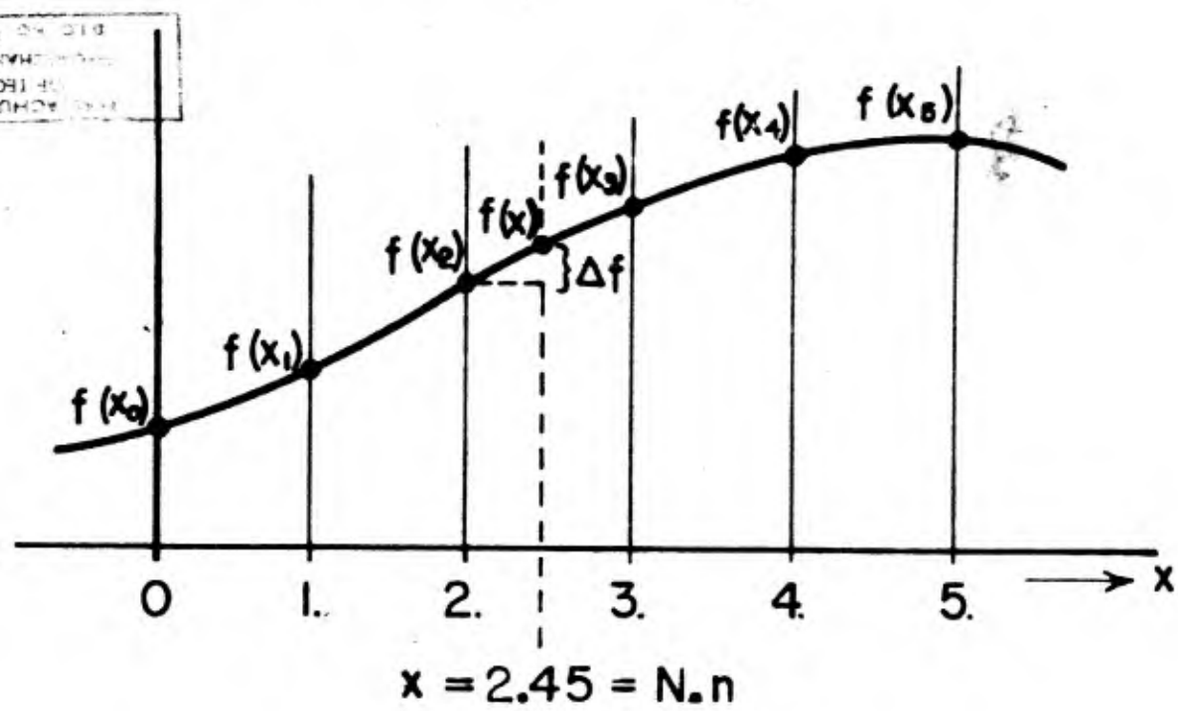
COLOUMB FRICTION SOLUTION

$$M\frac{dx^2}{dt^2} = P(t) + F(t) = f(t)$$

COMPARE
- $V > 0$ → $f(t) = P - F_{max}$
- $V \leq 0$ → ADD $V + I = V_a$

COMPARE
- $V_a \leq 0$ → $f(t) = P + F_{max}$
- $V_a > 0$ → $V = 0$

COMPARE
- $P > 0$ → $B = P - F_{max}$
  - COMPARE
    - $B > 0$ → $f(t) = P - F_{max}$
    - $B \leq 0$ → $f(t) = 0$
- $P \leq 0$ → $B = P + F_{max}$
  - COMPARE
    - $B > 0$ → $f(t) = 0$
    - $B \leq 0$ → $f(t) = P + F_{max}$



REGISTER	QUANTITY
S 27	$f(x_0)$
S 28	$f(x_1)$
S 29	$f(x_2)$
S 30	$f(x_3)$
S 31	$f(x_4)$
S 32	$f(x_5)$
S 53	n

OPERATIONS

S41 t ——— to AC

S42 t ——— to AR-SUB.

S43 t S 53 to AR-MULT.

S44 t ——— to AR-ADD

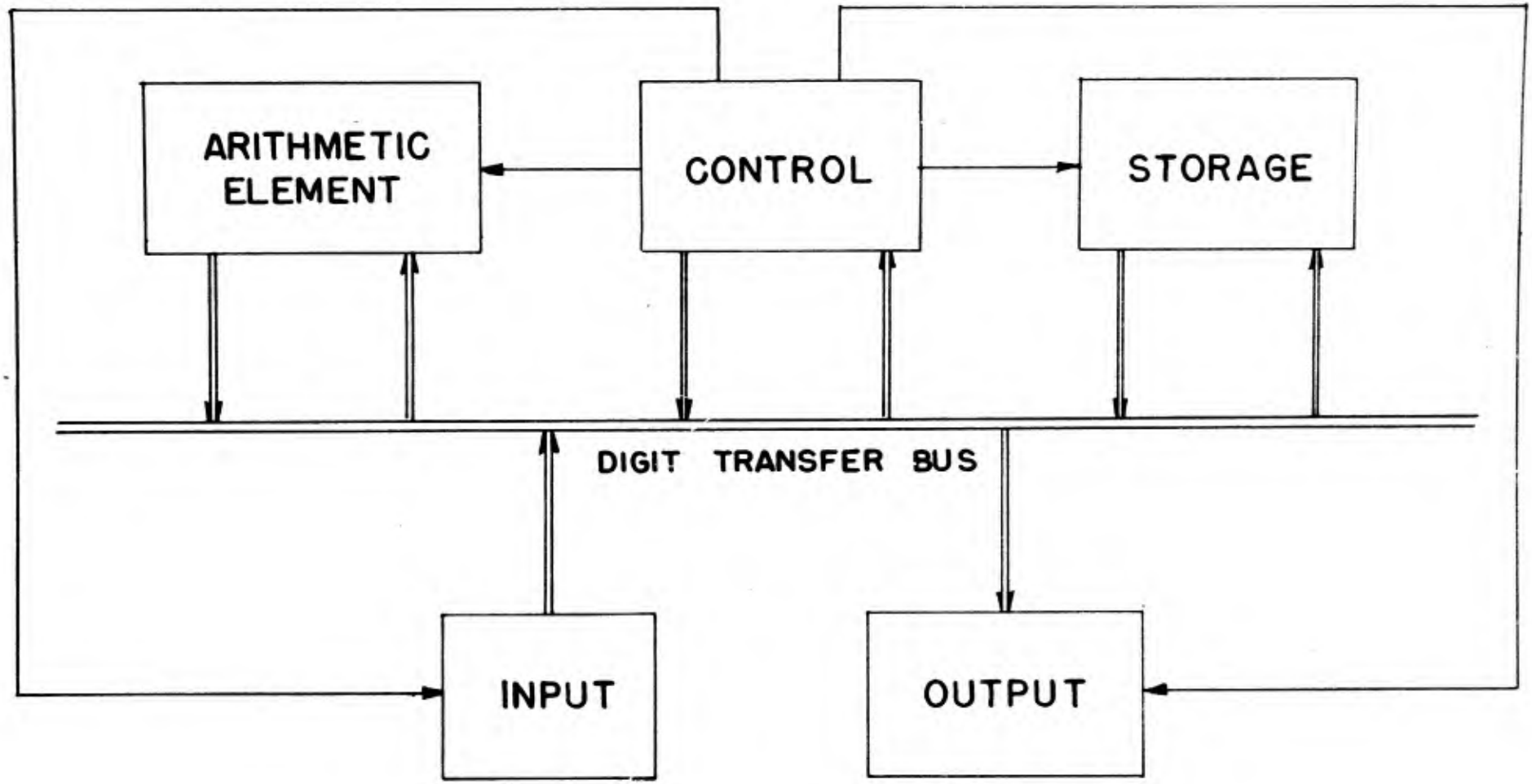
SEPARATE N FROM n

STORE n IN S53 ADD 27 to N

PLACE (27+N) IN S42 & S44 BLANKS

PLACE (27+N+1) IN S41 BLANK A-30352

A-3033^-1

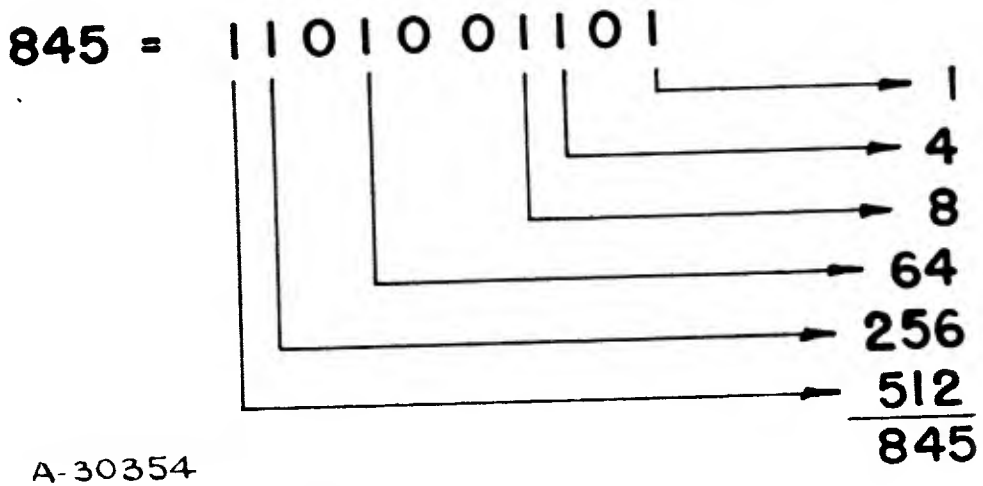


6345 RLK.
PRE

A-30339-1

BINARY NOTATION

DECIMAL	BINARY
0	0
1	1
2	10
3	11
4	100
8	1000
9	1001



A-30354

BINARY NOTATION

REPRESENTS POWERS OF 2

MULTIPLICATION TABLE:

$1 \times 1 = 1$

$1 \times 0 = 0$

$0 \times 0 = 0$

ADDITION:

$1 + 1 = 10$

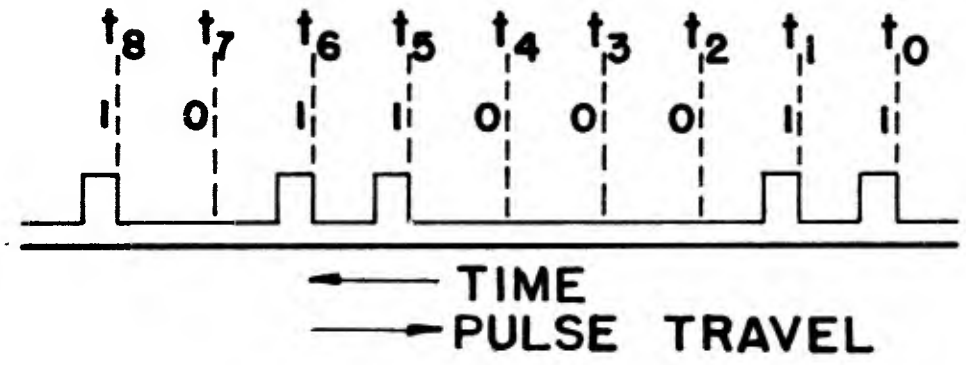
$1 + 0 = 1$

$0 + 0 = 0$

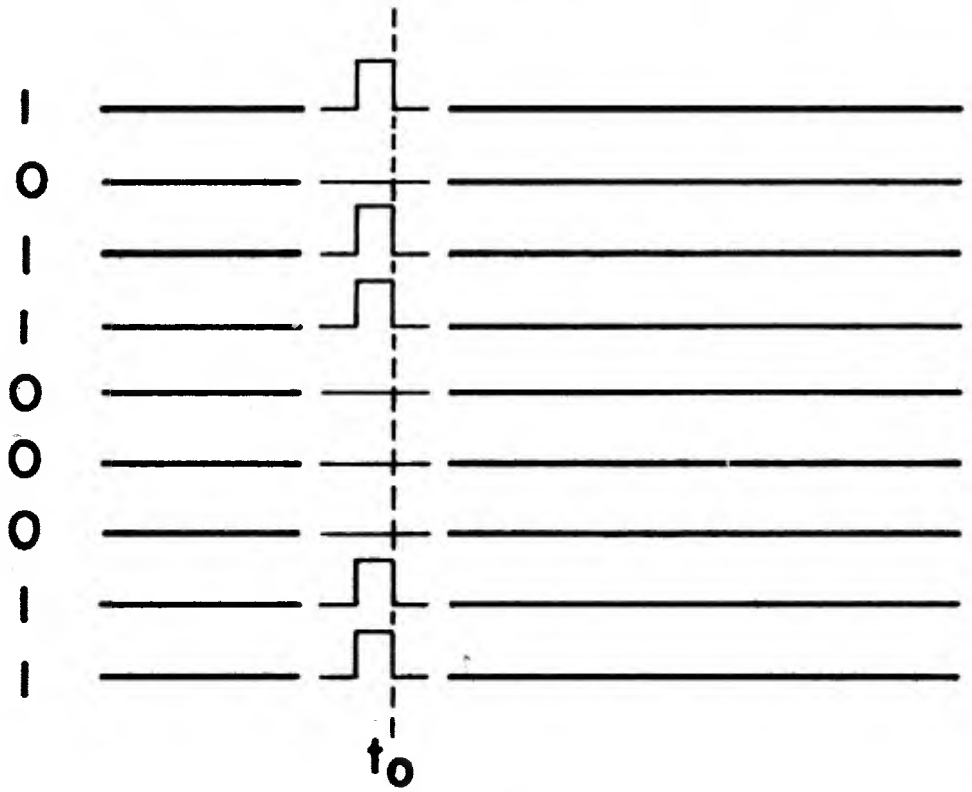
BINARY COLUMNS $\approx 3\frac{1}{3} \times$ DECIMAL COLUMNS

ONLY DIGITS 1 AND 0 REQUIRED IN EQUIPMENT

A-30355

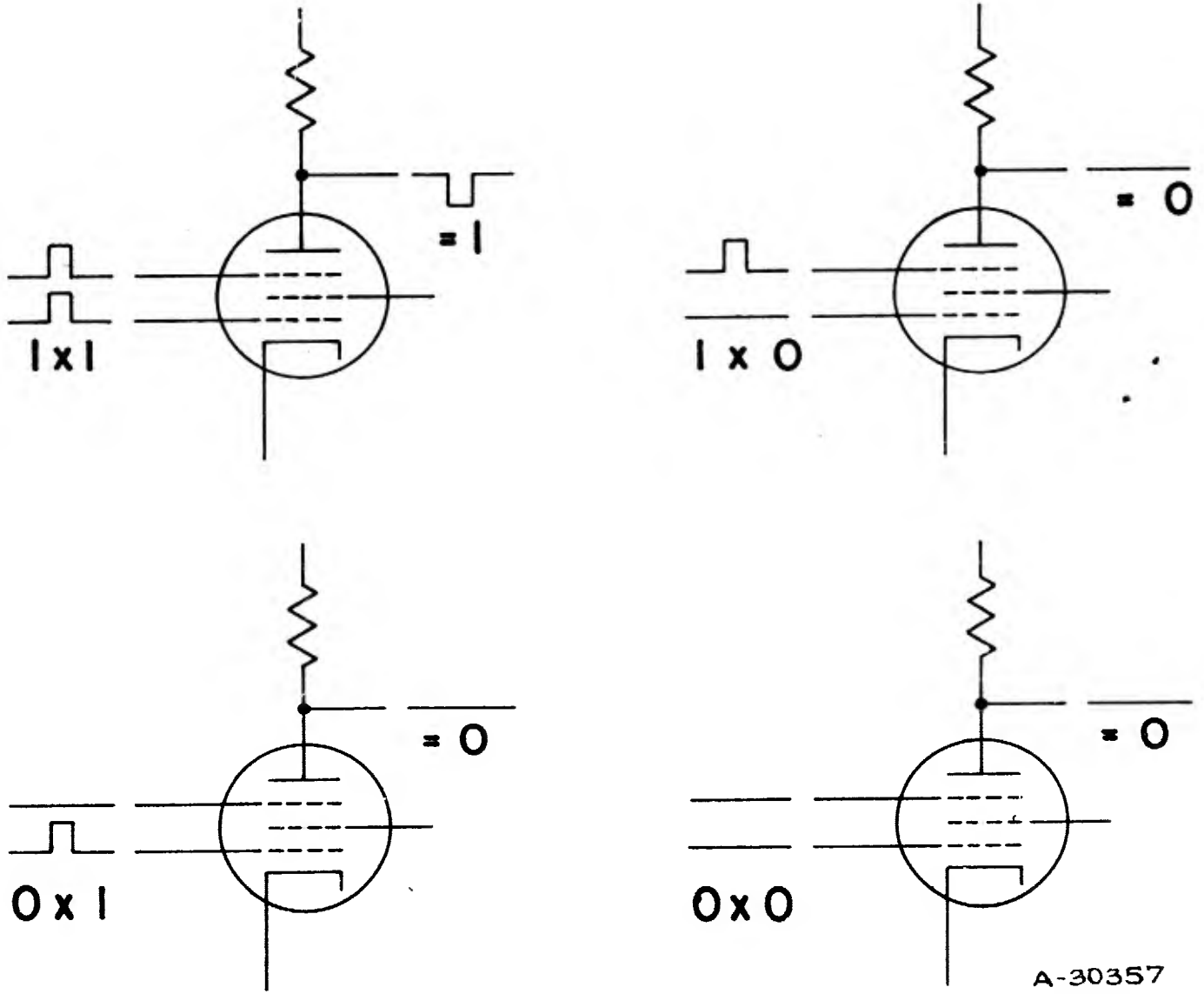


SERIAL DIGIT TRANSMISSION SINGLE LINE

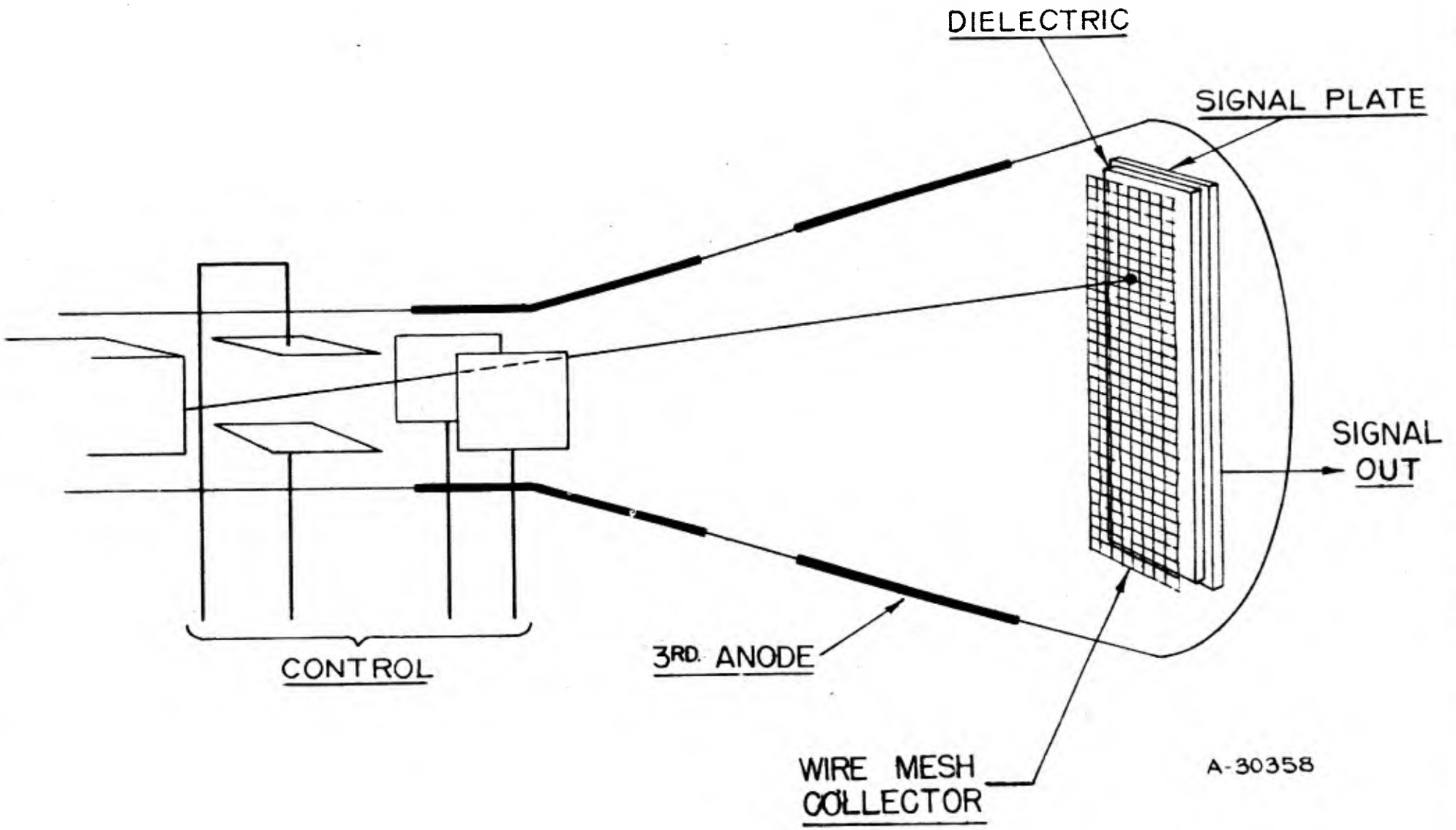


PARALLEL DIGIT TRANSMISSION MULTIPLE LINES

A-30356



A-30358-1



MASSACHUSETTS INSTITUTE OF TECHNOLOGY
SERVOMECHANISMS LABORATORY
D.I.C. NO. 6345

DR.

A-30358-1

COMPUTER	INTERNAL STORAGE TYPE	EQUIV. BINARY CAPACITY	SPEED MICROSECONDS PER MULTIPLICATION	CONTROL
HARVARD MK I	MECHANICAL	5,500	3,000,000	TAPE
BTL	RELAY			TAPE
J. OF P. ENIAC	FLIP-FLOP	660	3,000	PLUGBOARD
HARVARD MK II	RELAY	4,600	700,000	TAPE
J. OF P. EDVAC	MERCURY	15,000	1,000	WIRE
MIT WWI	ELECTROSTATIC	32,000	*	FILM
MIT WW2	ELECTROSTATIC	640,000	35	FILM
RCA IAS	SELECTRON	160,000	120	WIRE
BU. STDS		170,000	3,000	

A-30359

MEMORANDUM NO. M-63

Written by: Robert R. Everett

Subject: Digital Computing Machine Logic

Date: March 19, 1947

Illustrations:

A-30339 A-30403
A-30347 A-30365
A-30394 A-30409
A-30395 A-30405
A-30396 A-30406
A-30354 A-30392
A-30397 A-30393
A-30398 A-30391
A-30399 A-30415
A-30400 A-30419
A-30401 A-30414
A-30402 A-30418
A-30417 A-30501
A-30410 A-30500
A-30502

This seminar will be concerned with the parallel digit transmission type of electronic computer described by Mr. Forrester in last week's seminar. Illustration A-30339 shows a simplified block diagram for such a computer. The five elements are the input, output, storage, control, and arithmetic element. The input is needed to transfer both numbers and orders to the computer while the output is used for extracting results. All the numbers, initial conditions, partial results and orders are stored in the storage element. No distinction is made between orders and numbers. A single arithmetic element is provided, the abilities of which will be described later. The control directs the extraction of orders from the storage and sets up the proper switching to carry out those orders.

A list of basic computer operations is given in A-30347. The arithmetic element will be able to carry out the basic operations of addition, subtraction, multiplication and division. As will be shown later, it is not necessary to actually build in the division process since division can be performed by using the other basic orders.

In addition to these basic arithmetic operations are the important control operations of sub-programming, comparison, and substitution. Normally, the control extracts orders in sequence from storage. The sub-program order allows changing this sequence. By means of this order the control may be instructed to take as the next order any order stored in any register. The comparison order is similar to sub-programming except the decision as to whether or not to change the program is made dependent upon the sign of the number in the arithmetic element. If this number is negative, the control will ignore the comparison order and proceed in the previous sequence. If the number in the arithmetic element is positive or zero, the control will be shifted to a new position. The substitution order allows modifying orders previously in storage by substituting into them results which have been calculated.

Illustration A-30394 illustrates a basic order. The orders are placed in storage in binary code form. The order consists of two parts: an operation section which decides the operation to be carried out and a storage register number which chooses that piece of information in the storage that is to be transferred.

The sizes shown are for the WHIRLWIND II computer. Since WHIRLWIND II will have a register length of forty digits, two of these orders can be stored in a single register. The order used in WHIRLWIND I will be of the same type except that only five digits will be allowed for the operation and eleven digits for the storage number since only two thousand registers are to be provided. It is likely that for mechanical convenience the operation and storage register number sections will be stored in opposite order; that is, the left section will be the register number, while the right section will be the operation.

A possible order code is shown in Illustration A-30348. This is only one of many possible codes and does not show such special orders as input and output controls, operations using absolute values, and others which will be built into the computer.

The order ga clears the accumulator in the arithmetic element and transfers the number held in the storage register number section of the register into the accumulator.

The order ad transfers the number in the storage register corresponding to its register number into the arithmetic element and adds it to whatever number was previously held by the accumulator.

The order cs is the same as the order ga except the negative of the number is read into the accumulator. The order sm is the same as ad except that once again the negative of the number is read into the accumulator resulting as a subtraction instead of an addition.

The order mr multiplies the contents of the accumulator by the number in the corresponding register position. The order ts transfers the contents of the arithmetic element to storage and places them in the corresponding register position.

The order dv divides the number by the contents of the accumulator. The shift left and shift right orders simply multiply the contents of the accumulator by powers of two. The sp, cp, and id orders have been briefly described above.

WHIRLWIND computers will use the base 2. However, it will not be necessary for the people who use the computer to do any calculation in this base. Special devices will make the conversions from decimal to binary notation for the input and from binary to decimal notation for the output.

Any integer greater than one may be used as a base for computation. The customary decimal base is an outgrowth of the fact that we have ten fingers. The base 12 has been suggested because 12 is divisible by more integers than is 10. The base 2 is convenient for computer calculation because only two digits are necessary which means that we can use any computing elements which have but two stable states. Illustration A-30395 shows briefly the base system of representing numbers. A number written in any

base appears the same as a number written in any other except for possible new integers needed. Possible numbers extend in either direction with any desired power of the base. On the scale shown, numbers which are exceedingly small will require digits which are far to the right, while numbers which are exceedingly large will require digits which are far to the left. Because numbers can extend indefinitely in either direction, this number notation can be called a number spectrum. Any rational number may be written in any base system. Illustration A-30396 shows equal numbers written in several bases. The base 27 has actually been proposed for certain computing problems where a large base is needed, because the 26 letters of the alphabet are familiar to everyone. The 27th digit is simply a blank. Numbers written in this base 27 would require much less space than is our usual decimal base. The words of the English language can be considered as numbers written in the base 27. The English language uses the number of possibilities very inefficiently. The approximately half million existing English words could all be represented by four-letter words. Basic English which only has 650 words could be entirely represented by simply two-letter words. However, some of these words would be very difficult to pronounce.

Illustration A-30354 shows the relation between binary and decimal notation.

As stated above, a number spectrum extends indefinitely in both directions so that if any desired number is to be represented, an infinite register capacity is needed. Since only a finite register length is available on any physical machine, the problem of just which digits to select is of great importance. Illustration A-30397 illustrates several possibilities. The computer can consider, for instance, only numbers which are considerably larger than one, very near 1, or much less than 1. Different problems arise in these different cases. In Illustration A-30398 the effect of a multiplication is shown. A number considerably larger than one, when squared or multiplied by a number of equal magnitude, becomes much larger. If only a finite register length is available, the position of the register on the number spectrum must be shifted to the left. For a number much less than 1, a squaring produces a result which requires transferring the register position to the right. Where the number is very near 1, the squaring process does not appreciably move the register on the spectrum. For this reason it is convenient to consider the numbers used by the computer as being very near 1. For WHIRLWIND computers the decimal point is to be considered at the left-hand end of the register.

Since naturally we must in the course of general computations consider numbers which may vary over a very wide range, we must associate a scale factor with each number. The method of handling scale factors is determined by whether we are using a fixed or floating point system in the machine. In Illustration A-30399 consider a number which for convenience we will assume is larger than 1. Assume a machine having its point at the left end of the register. Associated with this number must be a scale distance between the machine point and the true point. Now, if upon further computation this quantity becomes much smaller, we have available two

possibilities. First, we can move the register to the right so that the number still fills the register and have the machine compute and record the reduced scale factor. This is called a floating point system because the machine point moves with respect to the true point. The second possibility is to keep the register fixed on the spectrum and allow the number to fill only part of the register. The scale factor remains fixed. The wastage of the left-hand digits of the register in this latter system are compensated by the fact that it is no longer necessary for the machine to handle the scale factor.

It is now necessary to consider the methods of binary arithmetic in order to understand the processes which WWI and WWII will use for computing.

Consider the familiar problem of decimal addition. Illustration A-30400 shows a simple decimal addition problem. The digits are added in pairs, it being possible however to obtain a sum in any column which is greater than 9. The extra digit needed to describe this sum is known as a carry. This carry must be added in to the left-hand adjacent place. Ordinarily, this carry operation is accomplished without writing it out specifically as shown in the example. A decimal adding machine must be able to perform carry additions of this type.

Illustration A-30341 shows a binary addition. The two binary numbers shown are summed up and it is once again possible to obtain a sum in any column which requires two digits. Thus the sum of a 1 and a 1 in the binary system comes out 10, the 1 being a carry which once again must be added into the left-hand adjacent place. It is possible for these carries to produce other carries which must again be added in. In fact, it is possible, using 40-digit binary numbers, for a carry to be propogated the full length of the number requiring 39 carry additions in the process. It can be shown statistically, however, that the expected number of carries in a 40-digit addition will be about five. It is also possible to build binary adders which are able to sum the carry simultaneously in a single operation.

Illustration A-30403 shows a simple binary adder. The lower flip-flops have their input signals supplied to their cathodes. They form simple scale of two counters. The addition of two ones to a flip-flop will restore it to zero, the overflow setting one of the upper row of flip-flops. These upper flip-flops store the carries. Following the first operation of addition the sum will appear in the lower flip-flops and the carries in the upper flip-flops. Gate tubes are connected to these upper flip-flops in such a manner that the gate is open if the flip-flop holds a 1. If, then, the other grids of the gate tubes are pulsed, a carry pulse will be transmitted to the left-hand adjacent adder flip-flop. At each pulsing of the carry line, one carry addition will be performed.

Negative numbers must be handled. Several possibilities are shown in Illustration A-30417. The first is to handle the number as its absolute magnitude with an associated sign. This method is the customary

notation. From the point of view of the computer, however such a notation is not too satisfactory since it requires a special subtracting unit and the ability to discriminate signs. Methods more convenient from the point of view of the computer are those using complements. In the tens complement system, the positive number is subtracted from some higher power of the base than can be held in a standard register. The sum of a number and its complement is therefore this power of the base. The digit that signifies this power will lie off the left-hand end of the register the sum appearing as a zero. Since therefore we have obtained a number which when added to the original number equals zero, we have satisfied the requirements for a negative number designation. A similar method but more convenient is the use of nines complements. In this system the positive number is subtracted from a power of the base less 1. The convenience may be noted from the illustration in that the nines complement is obtained by subtracting each digit of the positive number from nine. In binary notation each digit of a positive number is subtracted from one. This is equivalent to replacing all zeros by ones and ones by zeros. The conversion from nines complement to tens complement can be effected by adding a one to the nines complement. Unfortunately, this addition may produce carries requiring an adder and an addition time for changing the sign of a number.

By using the complement system, an adder may be made to subtract. The 9's complement system avoids propogation of carries and is thus more suitable for a machine of the Whirlwind type. Subtraction processes are shown in Illustration A-30410. The first example shows a subtraction in which the result is negative. The subtrahend is complemented and added to the minuend. The result is the difference in its complemented form. The second example shows the subtraction of two numbers whose difference is positive. Once again the subtrahend is complemented and added to the minuend. In this particular case, however, the difference is not in the desired form. It consists of the desired difference $N_1 - N_2$ plus the terms $2^n - 1$. Fortunately, the correction for these terms is easy. The 2^n represents a digit off the left-hand end of the register. If, therefore, this digit is carried around and added into the rightmost place, both the 2^n and the -1 will be corrected. This process is known as "end around carry" and is easily executed physically by connecting the carry section of the left digit of the accumulator to the right digit counter.

Multiplication is nothing more than the process of successive additions, the multiplicand being added to the partial product according to the digits in the multiplier. Illustration A-30404 shows a decimal multiplication example. The rightmost digit of the multiplier is multiplied by the digits of the multiplicand in turn. It will be noted that the results of each of these small multiplications may be a two-digit number requiring, therefore, an addition process to obtain the partial product. Following the use of the rightmost digit of the multiplier, the next digit to the left is used. The partial products formed are added together. Addition of partial products is continued until all digits of the multiplier have been used.

In multiplication, binary notation presents many advantages.

As shown in Illustration A-30385 the binary multiplication table is reduced to 1 x 1 = 1. Since the largest product of any two binary digits is the single binary digit one, there can be no carry between members in forming the partial product.

Illustration A-30409 shows a binary multiplication example. It is carried out in the same fashion as the decimal example previously discussed except that each step is now simply a choice as to whether or not to add in the multiplicand. This choice is governed by whether or not the multiplier digit is a one.

A modification of the binary multiplication process is shown in Illustration A-30404. This modification is helpful for the mechanization of the process. The multiplication begins as before with the addition of the multiplicand depending on the rightmost digit of the multiplier. Now, however, consider shifting both the multiplier and the partial product to the right one digit. The second step of the multiplication now considers the rightmost digit of the remaining part of the multiplier, the multiplicand being added in directly. It is not necessary to shift the multiplicand since the partial product has already been shifted. This process of shifting multiplier and partial product to the right and then adding in the multiplicand if the rightmost digit of the multiplier is a one, continues until the entire multiplier has been used.

Illustration A-30405 shows schematically a binary multiplier using this process. The multiplicand is stored in register AR and the multiplier in register BR. The partial product is built up in the adding unit AC. A set of gate tubes is used between AR and AC for adding in the multiplicand. If the rightmost digit in BR is a one, the multiplicand is added in. If it is a zero, the multiplicand is not added in. Following each addition the contents of AC and BR are both shifted to the right one digit. This places the desired digit of the multiplier at the right end of BR and provides the proper alignment of the partial product. Since in shifting BR to the right its leftmost digit becomes vacant, the rightmost digit of AC which would otherwise be shifted off the register and lost can instead be shifted into BR. When the multiplication process is completed the multiplier will have disappeared and the entire double-length product will appear in AC and BR.

Since the product of two n-digit numbers will result in a 2n-digit product, some method must be provided for handling those extra digits. In general, they are insignificant and can be thrown away. This process requires the proper rounding off of the number. If the last n digits are merely thrown away, the remainder will always be less than the complete number. In the course of a problem requiring a very large number of multiplications, the bias due to this procedure will become excessive. It is desirable to increase by one the rightmost digit of that part of the number which is kept if the number thrown away is greater than a half. A convenient way of accomplishing this in the decimal system is to add five to the first digit to be thrown away. If this digit is 5 or greater, a one will be carried to the rightmost digit of the saved number while, if this digit is less than

five, the right-hand digits will simply be discarded. This process is illustrated in Illustration A-30406. In binary notation an entirely similar process can be used in which a one is added to the first digit to be discarded resulting in a carry if this digit is a one and no carry if it is zero. No provision is made for properly caring for the ambiguous case in which the discarded part of the number is exactly one half. However, in 40-digit multiplication, 40 digits will be thrown away and the chances of this 40-digit number being exactly one half is very small. The bias due to this process is also very small and can in most cases be neglected.

The size of the numbers which can be handled by the computing machine is not limited by the machine register length. It is possible to use numbers which are any multiple of this register length. Illustration A-30392 shows this process for double-length numbers, that is, for numbers whose length is twice the number register length of the machine. The numbers are stored in two halves in two separate registers. Addition requires the separate summing of the corresponding halves. The sum of the two smaller sections may be greater than the register capacity. If so, the overflow represents a carry to the rightmost digit of the sum of the two larger sections. It is necessary to provide for handling this carry efficiently.

For multiplication the product can be formed by obtaining the four partial products resulting from multiplying the double-section numbers together section by section. Two peculiarities again arise. First, the product of the two small sections themselves is a small number which would ordinarily be lost in the rounding process. This product can therefore be omitted. Secondly, the two larger sections are assumed to have with them an associated register full of zeros. Thus, their product which will be double length as usual is actually entirely significant and both sections must be kept. The section which is usually rounded must be added to the smaller section of the result. It is once again necessary to take care of possible carries between sections.

The ability to handle double-length numbers will be important for WHIRLWIND I. The sixteen-digit capacity now proposed is equivalent to less than five decimal digits which is inadequate for most computing problems. By using double-length numbers, computations can be carried out using ten decimal places, thus greatly increasing the scope of the machine. WHIRLWIND II will also be able to handle double-length numbers for use in those problems requiring more than twelve decimal digits.

Equivalent methods may be applied for computing using triple-length numbers. Illustration A-30393 illustrates how a triple-length number product may be obtained. The number of orders required for computing using multiple-length numbers becomes quite large. Sub-programs can be used however.

It was mentioned earlier that it is not necessary to build division into the machine since it could be built up out of the more basic

arithmetic operations. This comment is also true of square rooting. A general method for obtaining reciprocals and roots is to use an iterative procedure based on the solution of an algebraic equation by Newton's method.

Illustration A-30391 shows Newton's method. The curve $y = f(x)$ intersects the axis at some point. It is desired to find this point by a process of successive approximations beginning with some original guess. Other methods are available but Newton's has the advantage of simplicity and speed.

Illustration A-30415 shows the application of Newton's method to performing the operation of division using only addition and multiplication. The equation $f(x) = a - \frac{1}{x}$ has a root at $x = \frac{1}{a}$. Applying Newton's method an equation $x_{n+1} = x_n(2 - ax_n)$ is obtained which will converge to $x_n = \frac{1}{a}$. This equation requires only additions and multiplications and results in the reciprocal. It converges very rapidly at each step. If the first guess is good to 10%, three iterations will give a result good to eight decimal places.

Illustration A-30419 shows the application of Newton's method to obtaining square roots. If the equation chosen is $f(x) = x^2 - a$, then x_n will converge to the $\sqrt{a}$. This process requires division and is satisfactory for a machine able to divide but will be quite lengthy for a machine that obtains division by iteration.

If the equation $f(x) = a - \frac{1}{x^2}$ is used, then x_n converges to $\frac{1}{\sqrt{a}}$ in a process requiring addition and multiplication only. It is quite easy to get the $\sqrt{a}$ from its reciprocal by simply multiplying by a itself.

Illustration A-30414 gives formulas by analogous methods for obtaining the p^{th} root where p is any integer.

It was mentioned earlier that the machine itself will perform all the necessary conversions for decimal to binary notations and binary to decimal notations. Illustration A-30418 shows two methods for converting from decimal to binary notations. Equivalent methods exist for converting from binary to decimal notations.

The first method consists of using the binary equivalent for the ten decimal integers. The leftmost digit of the decimal number is converted to binary and then multiplied in a binary multiplier by the binary equivalent of ten. To this result is added the binary equivalent of the second decimal digit. This process is repeated until the entire number has

been converted. The second method uses a decimal computer instead of a binary computer and consists of successively dividing the decimal number by two and noting the remainders of each step.

The power and flexibility of digital computing machines becomes most apparent in those instances where the computer carries the burden of generating its own controlling program. It is often customary for the mathematician to indicate in compact symbolic form the answer to a numerical process. The determinant and matrix notations are examples of this compact nomenclature.

A generalized program sufficient for multiplication of any two matrices will now be discussed. In this generalized program, 47 orders are sufficient to control any matrix multiplication. The program can be kept in a library for use as required.

As an example of programming methods and of the general power of a computing machine of this type, a generalized program for matrix multiplication will be discussed. This program will provide for the multiplication of any two matrices, not necessarily square, which can be multiplied within the capacity of the machine. Once written and put on a tape the program will be usable as often as desired. It will only be necessary to insert the elements of the matrices, certain information as to their dimensions and location in storage, and start the machine. The program can also be used as part of a larger program of which matrix multiplication is one of the desired operations.

There is nothing unusual about this program or the methods used. A program of this type can be written for a vast number of arithmetical processes including inverting a matrix, solving a set of simultaneous linear algebraic equations, a set of simultaneous linear differential equations, algebraic equations of high order, etc., once the method of solution has been determined.

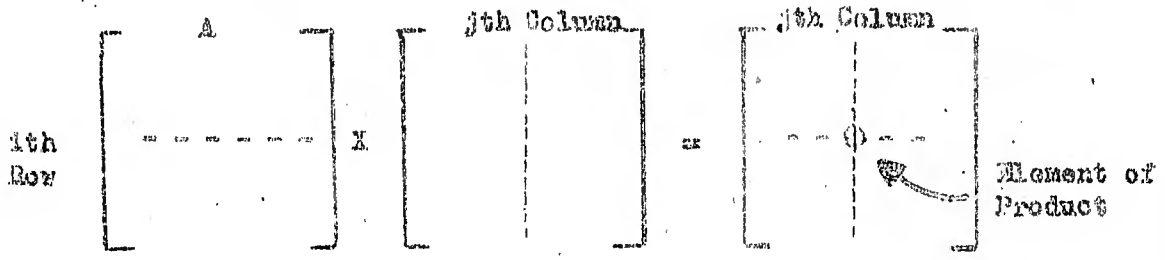
A matrix is a rectangular array of elements which obeys certain rules for multiplication and addition. A typical matrix is:

$$\begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & & & \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix}$$

The process of matrix multiplication is as follows. To obtain the ith element of the jth column of the product AB = C, select the ith row of A and the jth column of B, and sum the products of their corresponding elements, beginning at the left-hand end and the top, respectively. Thus,

$$C_{ij} = \sum_{r=1}^{n} a_{ir} \times b_{rj}$$

Where a, b, and c are elements of matrices A, B, and C, respectively.



The product of two, 2x2 matrices

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \times \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} (a_{11}b_{11} + a_{12}b_{21}) & (a_{11}b_{12} + a_{12}b_{22}) \\ (a_{21}b_{11} + a_{22}b_{21}) & (a_{21}b_{12} + a_{22}b_{22}) \end{bmatrix}$$

Before two matrices can be multipled together they must be conformable. That is, the number of columns in the first must equal the number of rows in the second.

The general process of multiplying two matrices becomes then, the following:

1. To obtain the first element of C multiply the first row of A by the first column of B.

   a) Multiply the first element of A by the first element of B.

   b) Multiply the second element of the first row of A by the second element of the first column of B.

   c) Add the two products.

   d) Continue until all the elements of the first row of A(and at the same time all the elements of the first column of B) have been used. The resulting sum is the first element of C.

2. To complete the first row in C, continue multiplying the first row in A by the succeeding columns of B.

   a) Repeat step 1, above, using the first row in A again but the second column in B. The result will be the second element in the first row in C.

   b) Continue, using the succeeding columns of B until all columns have been used. The resulting set of numbers will be the first row in C.

3. To complete the second row in C, repeat steps 1 and 2, but use the second row in A.

4. To complete the product matrix, repeat step 3 until all the rows of A have been used.

From the above discussion we can proceed to write the program required to put this procedure on the computing machine. The process is shown in diagramatic form in Illustration A-30501, to include the comparison orders the computer needs to determine when it has finished an element, a row, and the complete product.

The program is written out below. It follows the general process and diagram described. The changes in the program necessary to progress to the succeeding elements, rows, and columns, are made by bringing the actual orders into the arithmetic element and adding and subtracting numbers from them. The orders are restored in their modified condition so that the four basic orders that perform $\sum$ a b are used over and over again to perform all the separate $\sum$ a b's needed for the multiplication.

The symbols used in this program are discussed earlier in this memorandum.

The matrices to be multiplied are assumed to be stored in the computer storage in the following fashion: Matrix A is stored in sequence by rows. The elements of any row are stored in sequence from left to right, the groups of elements making up the rows being stored in sequence from top to bottom. Matrix B is stored in sequence by columns. The elements of any column are stored in sequence from top to bottom, the groups of elements making up the columns being stored in sequence from left to right. The order is merely convenient and not necessary. The program with slight modifications can be used for different orders of storage. It is not necessary that adjacent elements be in adjacent registers. The only requirement is that the storage be orderly and known; the given sequence is merely the simplest.

The elements of the product matrix C will be stored in sequence by rows as Matrix A since the elements will be obtained in this sequence.

The order of storage as well as other pertinent information is given in Illustration A-30500. The Program proceeds as follows. See pp. 1 - 2 for detailed orders and Illustration A-30402for progression diagram.

I. Start - Orders S1 to S6

The first step is to insert the initial register positions of the elements to be multiplied. This operation requires three transfers, one for each of the matrices concerned, or a total of six orders.

II. a · b - Orders S7 and S8

The multiplication of the two elements requires only two orders since the product is left in the accumulator.

III. Σ a b - Orders S9 and S10

The partial sum is brought into the arithmetic element and added to the new product. The result is stored in the register allotted to Σ a · b.

IV. Increase S(a) by 1 - Orders S 11 to S13

The next step is to increase the register number in Order S7 by 1 which will pick the next element in the row in Matrix A. This modification is made at this time because it effects a small saving in total number of orders and total number of operations. Three orders are required for this modification.

V. Check on Completion of an Element - Orders S14 and S15

An element of the product will be complete when all the elements of a row of the first matrix have been used in a sum. Checking on the completion of a row can be obtained by comparing the register position in Order S7 with the register position holding the last element in the row. The comparison can be made by subtracting $S(a_n)$ from S7. S7 has already been modified in IV to be one more than the last register used. Therefore, if S7 $\leqslant$ $S(a_n)$ then the row has not been completed while, if S7 $\geqslant$ $S(a_n)$ it has. The comparison requires only two orders, a subtraction and the comparison itself.

VI. Increase S(b) by 1 - Orders S16 to S19

If the element is not complete as determined by the comparison order, then it is necessary to modify S7 and S8 to take the next elements in the two matrices, return to II, and continue the process. S7 has already been modified in IV. Four orders are required to modify S8 and return the control to S7, the start of II.

VII. Store Element - Orders S20 to S21

If an element has been completed, it must be stored or sent to the output (orders S20-S21).

Next, it is necessary to clear the register holding $\sum a\,b$ so that it will be available for the next stage of calculation. One method is to subtract $\sum a\,b$ into the accumulator which already holds $\sum a\,b$. The resulting zero is stored in the $\sum a\,b$ register effectively clearing it(Orders S22-S23).

Next, the order S21 containing the register number where the element C has first been stored is modified so that the next element to be computed will be stored in an adjacent register(Orders S24, S25, S26).

VIII. Check on Completion of a Row in the Product - Orders S27 to S28

A row in the product will be completed if all the elements of Matrix B have been used. Therefore, compare S8 which contains the register number of the last element of Matrix B used with the register number less 1, $S(b_k)$, of the last element in Matrix B. This comparison is effected by means of a subtraction and a conditional sub-program order. If $S(b) \neq S(b_k)$, the last element used will be at the bottom of some column other than the last. $S(b_k)$ must be 1 less than the register number of the last element in Matrix B because the equal case is lumped with the negative numbers in the conditional sub-program order.

IX. Return to Start of Row in Matrix A

Change to Next Column in Matrix B - Orders S30 to S36

If a row of the product is not complete, it is necessary to use the same row of Matrix A over again. Order S7 is modified by subtracting the number of elements in a row from its register position (Orders S30 to S32).

Changing to the next column of Matrix B is accomplished by simply taking the next element in line due to the order of storage(Orders S33 to S35).

The control is then returned to the start of II at S7 and the process repeated.

X. Check on Completion of Product - Orders S37 to S39

The product will be complete if all the elements of Matrix A have been used. The check can be made by comparing the register position in S7 with the register position holding the last element of Matrix A, $S(a_k)$. It is not necessary to subtract 1 from $S(a_k)$ since a 1 has already been added to S7.

The comparison is made with a subtraction and conditional sub-program order as before.

XI. Change to Next Row in Matrix A

Modify S(a_n)

Return to Beginning of Matrix B - Orders S40 to S45)

If the product is not complete, the next row of Matrix A is to be used. No modifications are necessary since the 1 added to S(a) in IV will take the first element of the next row due to the order of storage.

Since a new row is being used, it is necessary to change S(a_n) so that it is the register number holding the last element in the new row. This modification is accomplished by adding the row length to S(a_n) - (Orders S40 to S42).

Returning to the beginning of Matrix B is accomplished by subtracting the number of elements in Matrix B less 1 from the register position S(b) in Order S3.

The control is then returned to S7 as before.

XII. Stop - Order S47

If the product is complete, the machine is stopped.

MANUAL MULTIPLICATION

	Register No.	Contents	Description
I	S1	ca S55	Initial S(a) to AC
	S2	td S7	Initial S(a) to S7
	S3	ca S56	Initial S(b) to AC
	S4	td S8	Initial S(b) to S8
	S5	ca S57	Initial S(c) to AC
	S6	td S21	Initial S(c) to S21
II	S7	ca S(a)	a to AC
	S8	xr S(b)	Form a'b
III	S9	ad S48	Σ a b
	S10	ts S43	Store Σ a b
IV	S11	ca S7	Order S7 to AC
	S12	ad S49	Add 1 to S(a)
	S13	ts S7	Restore S7
V	S14	Sm S20	Form S(a) - S(a_n)
	S15	cp S20	Compare
	If S(a) $\leq$ S(a_n)		
VI	S16	ca S8	Order S8 to AC
	S17	ad S49	Add 1 to S(b)
	S18	ts S8	Restore S8
	S19	cp S7	Return control to S7

MATRIX MULTIPLICATION (Cont'd.)

	Register No.	Contents	Description
		If $S(a) \geq S(a_n)$	
	S20	oa S40	$\sum$ a b to AC
	S21	ta S(c)	$\sum$ a b to Result
	S23	Su S48	Subtract $\sum$ a b to clear AC
VII	S23	ta S46	Clear $\sum$ a b
	S24	oa S21	Order 21 to AC
	S25	ad S49	Add 1 to S(c)
	S26	ta S21	Restore order 21
	S27	oa S2	Order 6 to AC
VIII	S28	Su S50	Form $S(b) - S(b_n)$
	S29	op S37	Compare
		If $S(b) \leq S(b_n)$	
	S30	oa S7	Order 37 to AC
	S31	Su S32	Return to start from order
	S32	ta S7	Restore order 37
IX	S33	oa S6	Order 36 to AC
	S34	ad S49	Add 1 to S(b)
	S35	ta S6	Restore order 36
	S36	up S7	Return control to 37

MATRIX MULTIPLICATION (Cont'd)

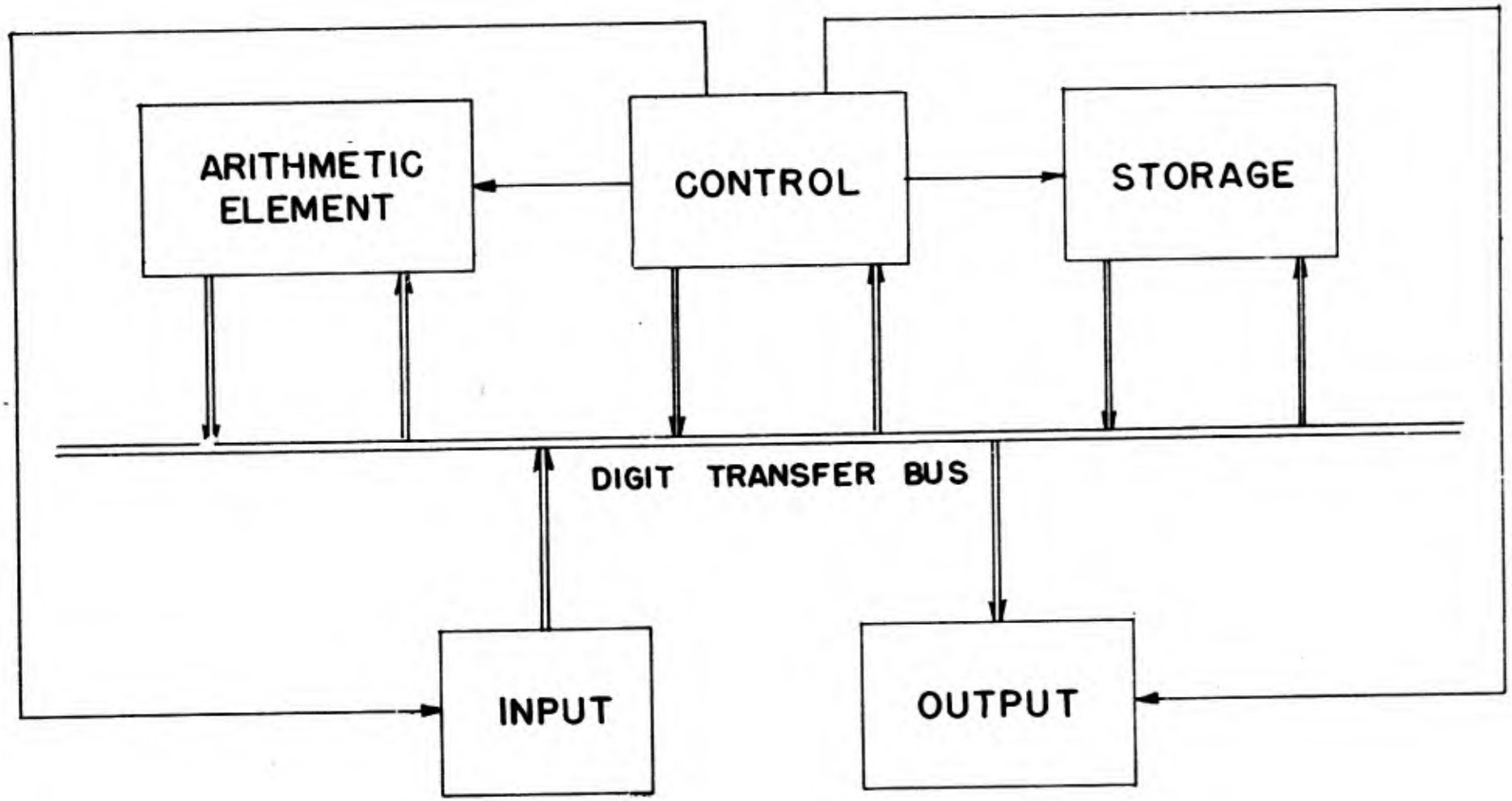
	Register No.	Contents	Description
		If $S(b) \geqslant S(b_k)$	
	S37	on S7	Order 7 to AC
X	S38	Su S53	Form $S(a) - S(a_k)$
	S39	op S47	Compare
		If $S(a) \leqslant S(a_k)$	
	S40	on S50	$S(a_n)$ to AC
	S41	a4 S52	Change $S(a_n)$ to end of next row
	S42	ts S50	Restore $S(a_n)$
	S43	on S8	Order S8 to AC
XI	S44	9x S54	Return to start of second matrix
	S45	ts S8	Restore order S8
	S46	9p S7	Return control to S7
		If $S(a) \geqslant S(a_k)$	
XII	S47	Stop	
	S48	$\frac{1}{2}$ a b	Start cleared
	S49	+ 1	
	S50	$S(a_n)$	Start at end of first row +1
	S51	$S(b_k)$	
	S52	Row length	
	S53	$S(b_k)$	
	S54	One less than the number of elements in Matrix B	
	S55	Initial S(a)	
	S56	Initial S(b)	
	S57	Initial S(c)	

Robert R. Everett

Robert R. Everett

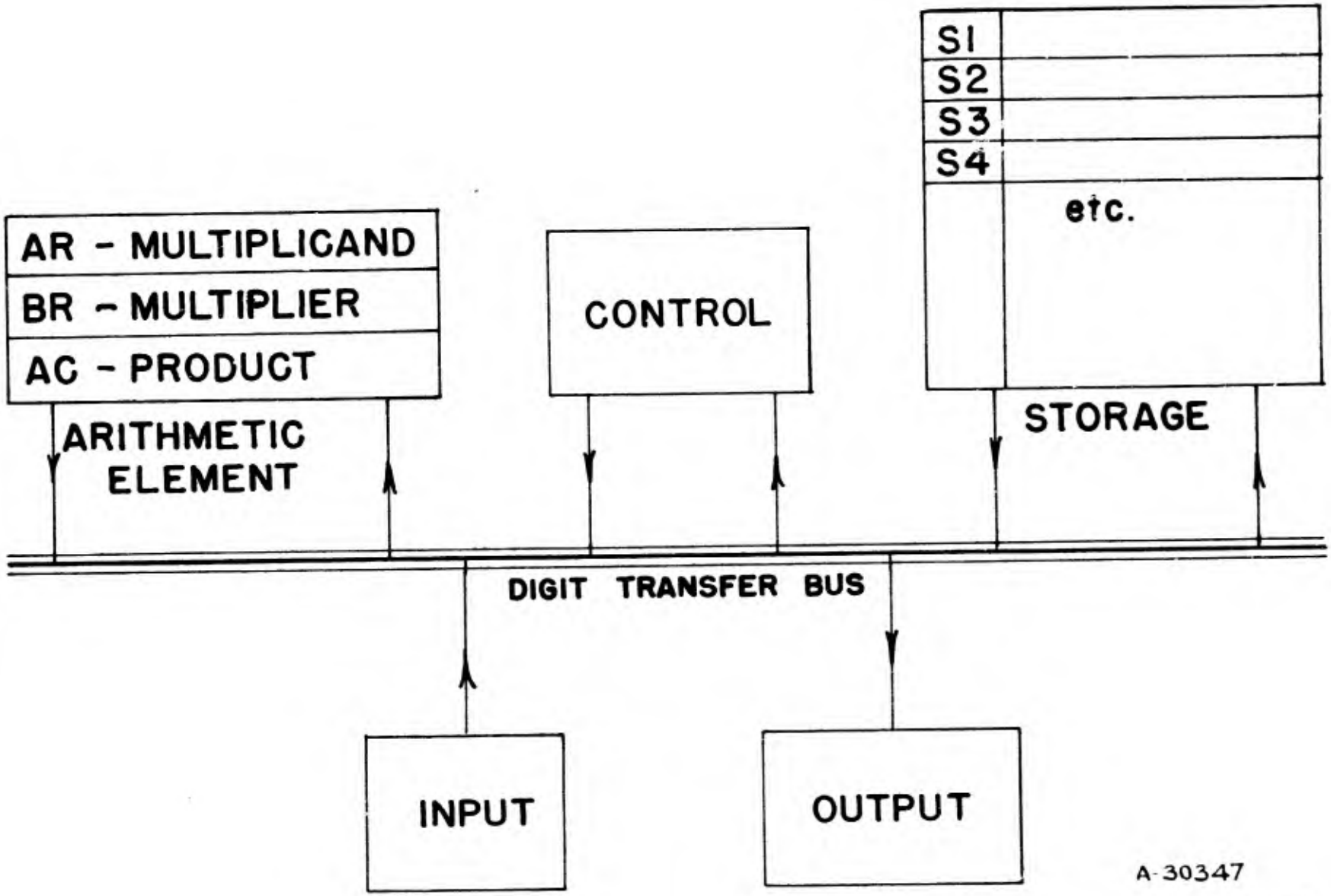
RRE:hcs

A-30339-1



6345 RLK.
RICE

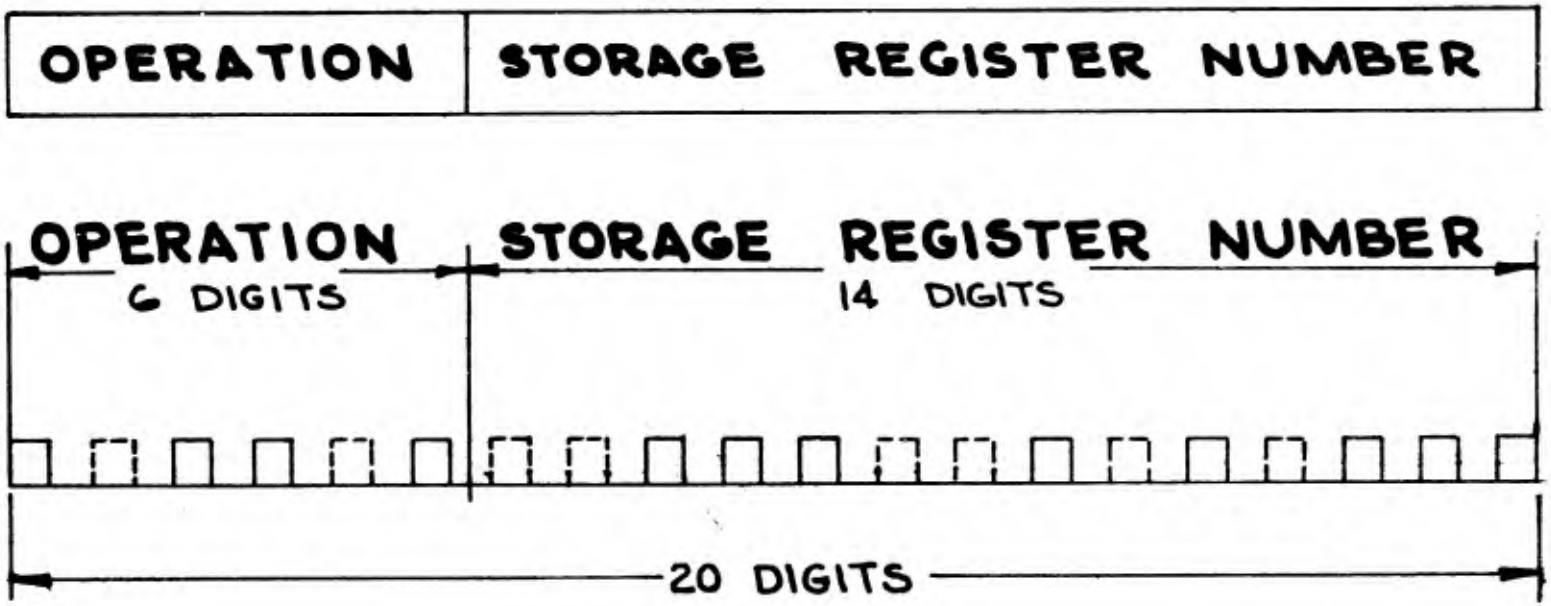
A-30339-1



A-30347

6345
RRG
3/20/47

A-30394



64 POSSIBLE OPERATIONS

16,384 POSSIBLE REGISTERS

1,048,576 POSSIBLE ORDERS

A-30394

G345
RRE
3/20/47

A-30395

NUMBER SPECTRUM

BASE 10 .... $a_2 \times 10^2 + a_1 \times 10^1 + a_0 \times 10^0 + a_{-1} \times 10^{-1} + \ldots$

BASE 12 .... $a_2 \times 12^2 + a_1 \times 12^1 + a_0 \times 12^0 + a_{-1} \times 12^{-1} + \ldots$

BASE 2 .... $a_2 \times 2^2 + a_1 \times 2^1 + a_0 \times 2^0 + a_{-1} \times 2^{-1} + \ldots$

BASE n .... $a_2 \times n^2 + a_1 \times n^1 + a_0 \times n^0 + a_{-1} \times n^{-1} + \ldots$

A-30395

NUMBER	EQUIVALENTS
DECIMAL	2234
BINARY	100010111010
BASE 27	CAT

A-30396

BINARY NOTATION

DECIMAL	BINARY
0	0
1	1
2	10
3	11
4	100
8	1000
9	1001

845 = 1 1 0 1 0 0 1 1 0 1

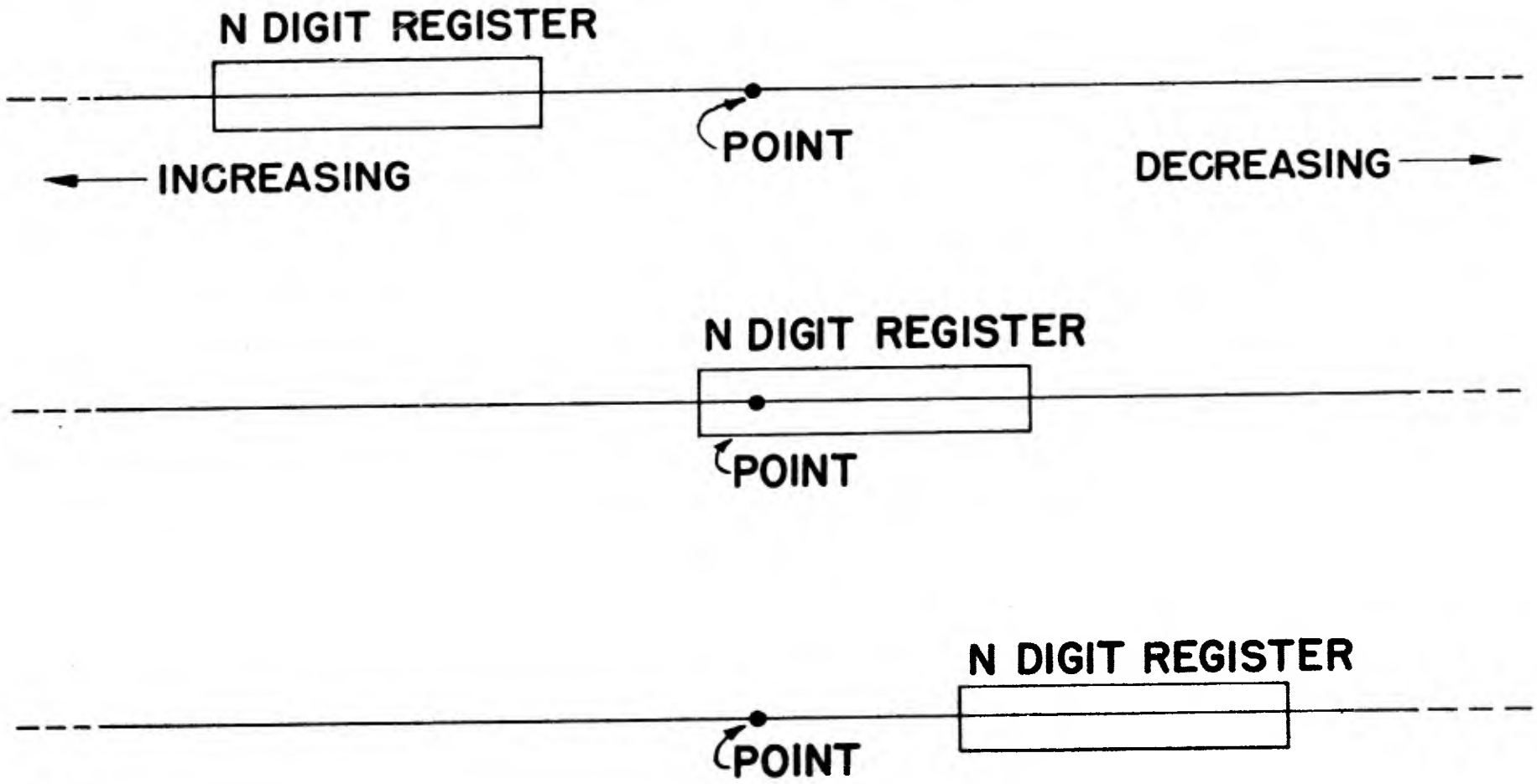
1
4
8
64
256
512
845

A-30354

6345
RRE
3/20/47

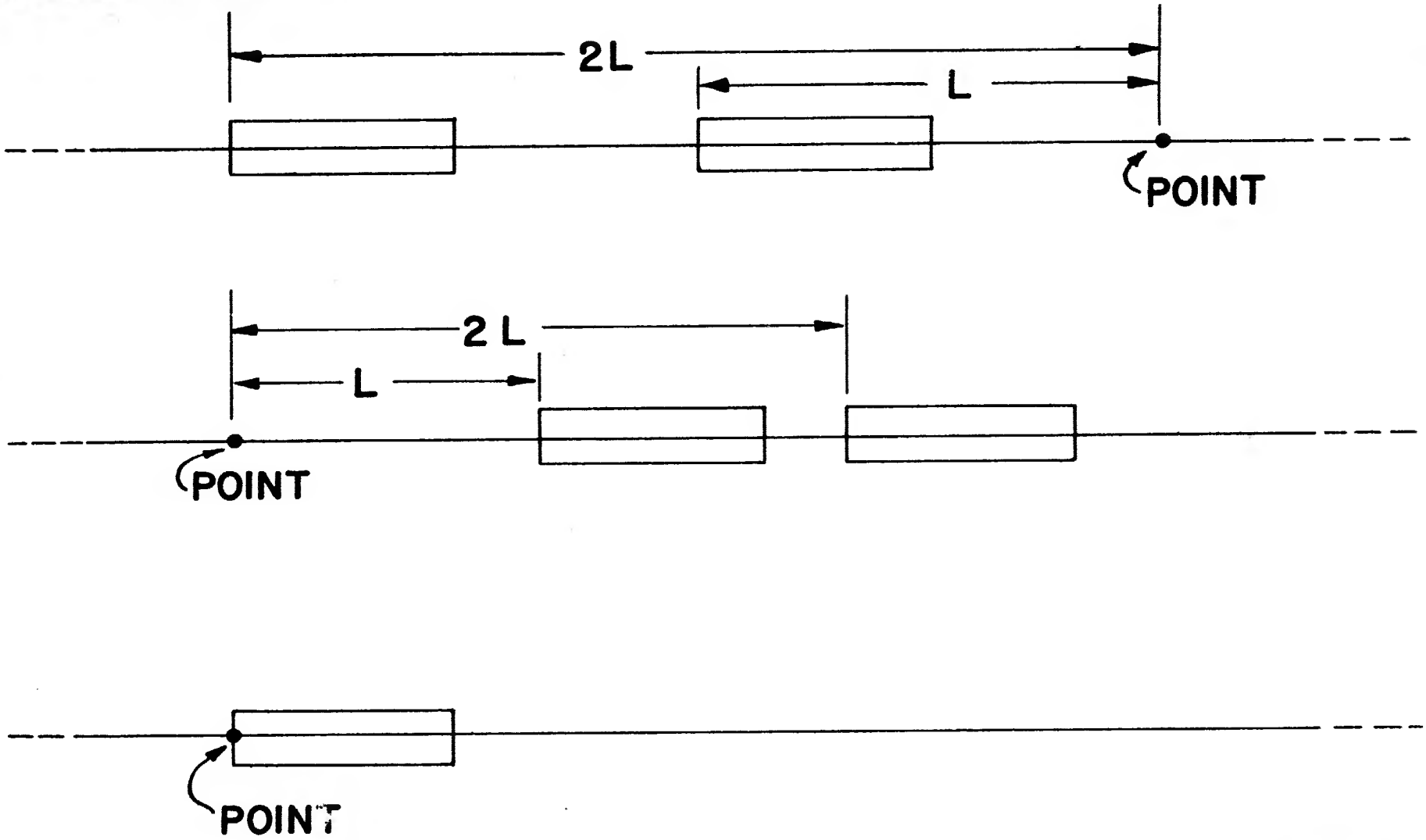
A-30397

REGISTER LENGTH



A-30397

REGISTER POSITION

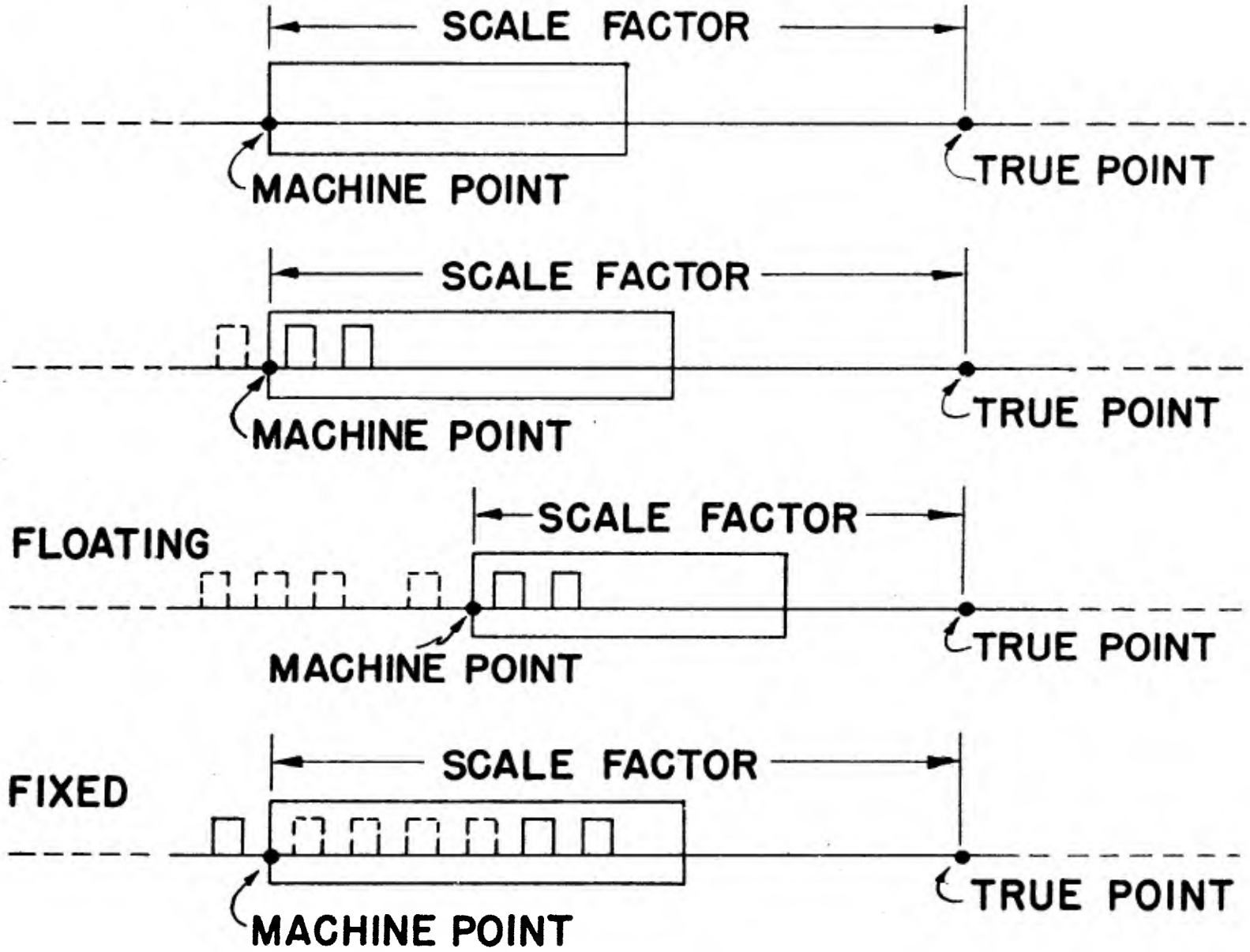


A-30398

C345
RRE
3/10/67

A-30399

FIXED AND FLOATING POINTS



A-30399

6345
RRE
3/20/47

A-30400

DECIMAL ADDITION

$$\begin{array}{r} 3546 \\ 1371 \\ \hline \\ 4817 \\ +1 \\ \hline 4917 \end{array}$$

A-30400

BINARY ADDITION

6345
RRE
3/20/47

A-30401

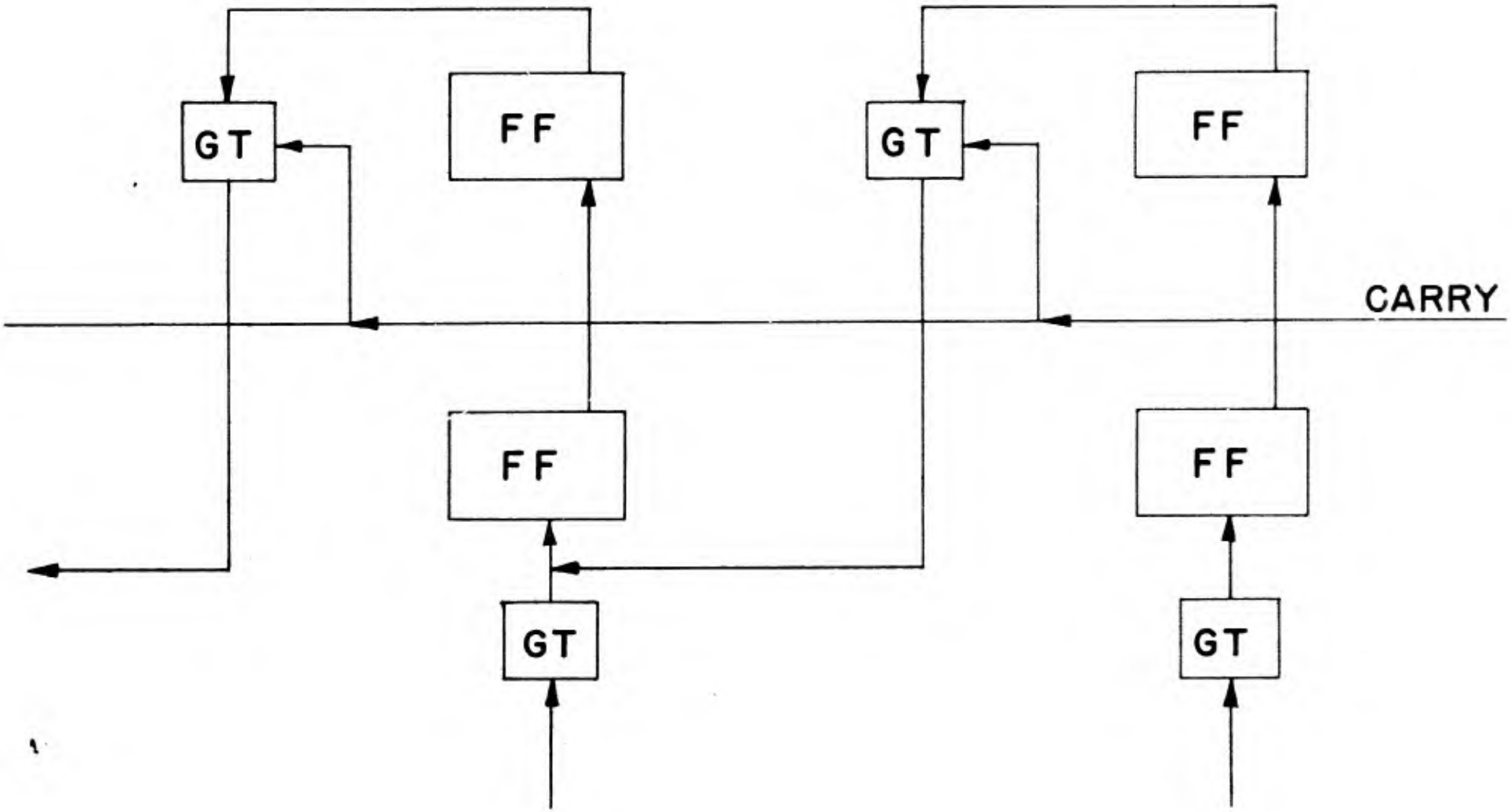
```
  11011010
  10110110
 ----------
  01101100
 ----------
  10100100
 ----------
 100000000
 ----------
 110010000
```

A-30401

6345
RRE
3/14/57

A-30402

A BINARY ADDER



A-30402

NEGATIVE NUMBERS

	DECIMAL	BINARY
POSITIVE NUMBER	223	11011111
NEGATIVE "	- 223	- 11011111
10's COMPLEMENT	1|777	1|00100001
9's "	1|776	1|00100000

A-30417

A-30410-1

SUBTRACTION USING 9'S COMPLEMENTS

BINARY			DECIMAL
+010110110	=	N_1	= 182
−011010010	=	$-N_2$	= −210
			−28
010110110	=	N_1	
100101101	=	$2^n - N_2 - 1$	
111100011	=	$2^n - (N_2 - N_1) - 1$	
−000011100	=	$N_1 - N_2$	= −28

G345 RKE 3/20/47

010110110	=	N_1	= 182
−010010010	=	N_2	= −146
			+36
010110110	=	N_1	
101101101	=	$2^n - N_2 - 1$	
1000100011	=	$N_1 - N_2 + 2^n - 1$	
END AROUND CARRY +1			
000100100	=	$N_1 - N_2$	= +36

6345
RRE
3/20/47

A-30403-1

DECIMAL MULTIPLICATION

356	MULTIPLICAND
627	MULTIPLIER
42	
35	
21	
2492	PARTIAL PRODUCT
12	
10	
6	
9612	PARTIAL PRODUCT
36	
30	
18	
223212	PRODUCT

A-30403-1

A-30355

BINARY NOTATION

REPRESENTS POWERS OF 2

MULTIPLICATION TABLE:

1 X 1 = 1

1 X 0 = 0

0 X 0 = 0

ADDITION:

1 + 1 = 10

1 + 0 = 1

0 + 0 = 0

BINARY COLUMNS $\approx 3\frac{1}{3}$ X DECIMAL COLUMNS

ONLY DIGITS 1 AND 0 REQUIRED IN EQUIPMENT

A-30355

6345 A 30355

A-30409-1

BINARY MULTIPLICATION & DECI. EQUIV.

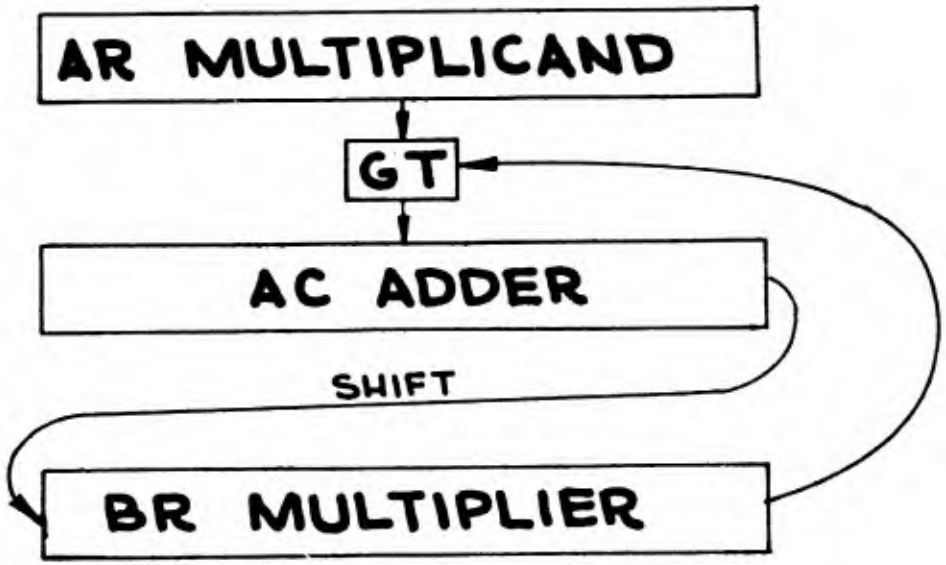
10110	MULTIPLICAND	22
10011	MULTIPLIER	19
		198
10110		22
10110		
1000010	PARTIAL PRODUCT	418
00000		
1000010	PARTIAL PRODUCT	
00000		
01000010	PARTIAL PRODUCT	
10110		
110100010	= 418 PRODUCT	

G345
RRE
3/23/67

6345
RRE
3/20/47

A-30405

BINARY MULTIPLIER



A-30405

6345
RRE
3/20/47

A-30406

ROUNDING OFF

DECIMAL

1734	6148
	5
1735	1

173	46148
	5
173	9

BINARY

11010	1010
	1
11011	

1101	01010
	1
1101	1

C345
RRE
3/20/47

A-30392

DOUBLE LENGTH NUMBERS

$$\begin{array}{rcl} A & + & B \times 2^{-n} \\ + \; C & + & D \times 2^{-n} \\ \hline (A+C) & + & (B+D) \times 2^{-n} \end{array}$$

POSSIBLE CARRY

$$\begin{array}{l} A + B \times 2^{-n} \\ C + D \times 2^{-n} \\ \hline AC + (AD + BC) \times 2^{-n} + BD \times 2^{-2n} \end{array}$$

POSSIBLE CARRY

DOUBLE LENGTH PRODUCT

A-30392

TRIPLE LENGTH NUMBERS

$$\frac{A + B \times 2^{-n} + C \times 2^{-2n}}{D + E \times 2^{-n} + F \times 2^{-2n}}$$

$$= AD + (BD + AE) \times 2^{-n} + (CD + BE + AF)\, 2^{-2n} + \ldots$$

POSSIBLE CARRIES

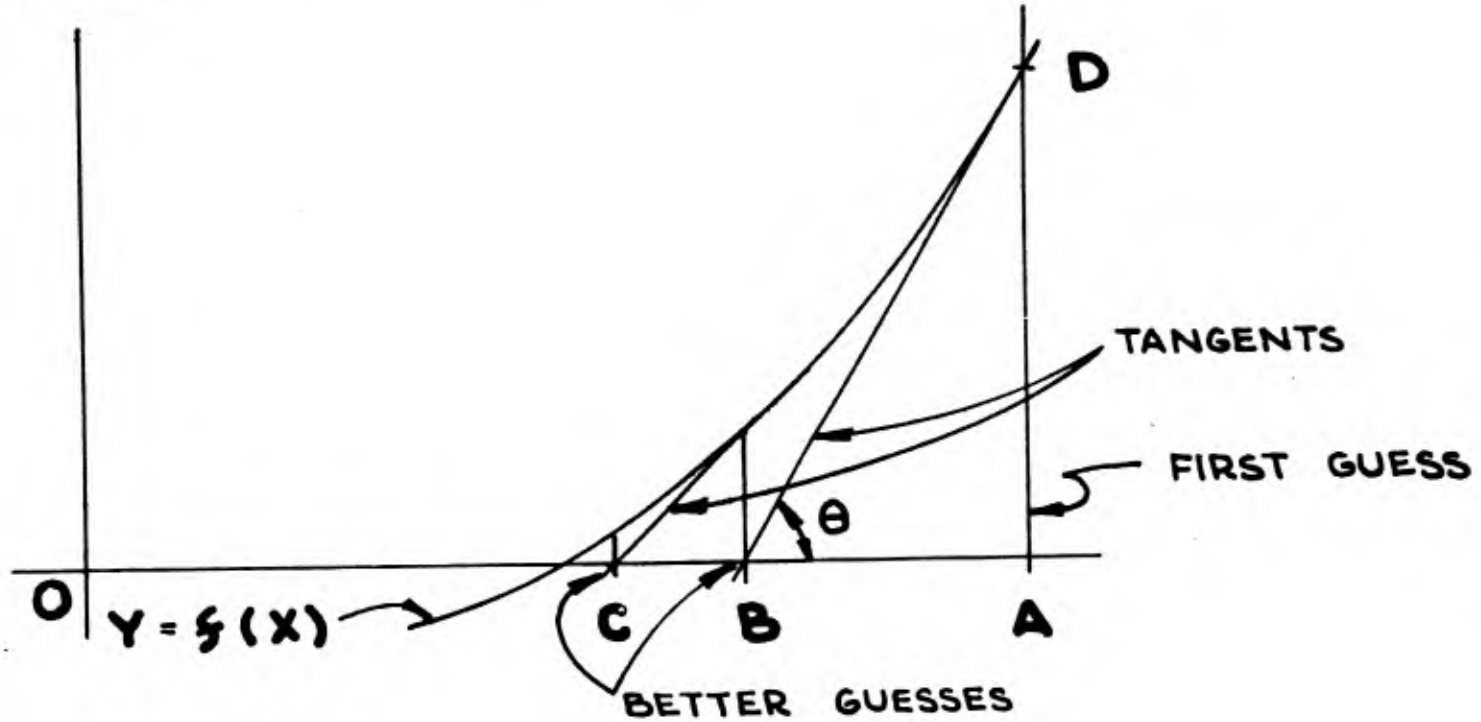
DOUBLE LENGTH NUMBERS

A-30393

6345
RRE
3/20/47

A-30391

NEWTON'S METHOD



$$X_0 = OA$$

$$\tan\theta = f'(X_0) = \frac{AD}{BA}$$

$$X_1 = OB = OA - BA \cdot X_0 - f(X_0)\cot\theta$$

$$= X_0 - \frac{f(X_0)}{f'(X_0)}$$

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)}$$

DIVISION BY ITERATION

$$X_{n+1} = X_n - \frac{f(X_n)}{f'(X_n)}$$

LET $f(X) = a - \frac{1}{X}$

$$f'(X) = \frac{1}{X^2}$$

$$X_{n+1} = X_n - \frac{a - \frac{1}{X_n}}{\frac{1}{X_n^2}}$$

$$= X_n - a X_n^2 + X_n$$

$$= X_n (2 - a X_n)$$

A 30415

SQUARE ROOT BY ITERATION

A-30419

G345
RAE
3/20/47

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

LET $f(x) = x^2 - a$

$f'(x) = 2x$

$$x_{n+1} = x_n - \frac{x_n^2 - a}{2x_n}$$

$$= x_n - \frac{x_n}{2} + \frac{a}{2x_n}$$

$$= \frac{1}{2}\left(x_n + \frac{a}{x_n}\right)$$

$x_n \longrightarrow \sqrt{a}$

$$f(x) = a - \frac{1}{x^2}$$

$$f'(x) = \frac{2}{x^3}$$

$$x_{n+1} = x_n - \frac{a - \frac{1}{x_n^2}}{\frac{2}{x_n^3}}$$

$$= x_n - \frac{a x_n^3}{2} + \frac{x_n}{2}$$

$$= \frac{x_n}{2}(3 - a x_n^2)$$

$x_n \longrightarrow \frac{1}{\sqrt{a}}$

A-30419

G345
RRE
3/20/47

A-30414

FORMULAS FOR THE P^{th} ROOT

RECIPROCAL P^{th} ROOT $\quad X_{n+1} = \frac{X_n}{P}\left[(P+1) - aX_n^P\right]$

$$X_n \longrightarrow a^{-\frac{1}{P}}$$

P^{th} ROOT $\quad X_{n+1} = X_n \frac{Pa + X_n^P}{a + PX_n^P}$

$$X_n \longrightarrow a^{\frac{1}{P}}$$

A-30414

A-30418

6345
RRE
3/20/47

DECIMAL TO BINARY CONVERSION

DECIMAL NUMBER 235

BINARY EQUIVALENT OF 2 = 10

" " " 10 = 1010 × 1010

" " " 20 = 10100

" " " 3 = 11 + 11

" " " 23 = 10111

" " " 10 = 1010 × 1010

" " " 230 = 11100110

" " " 5 = 101 + 101

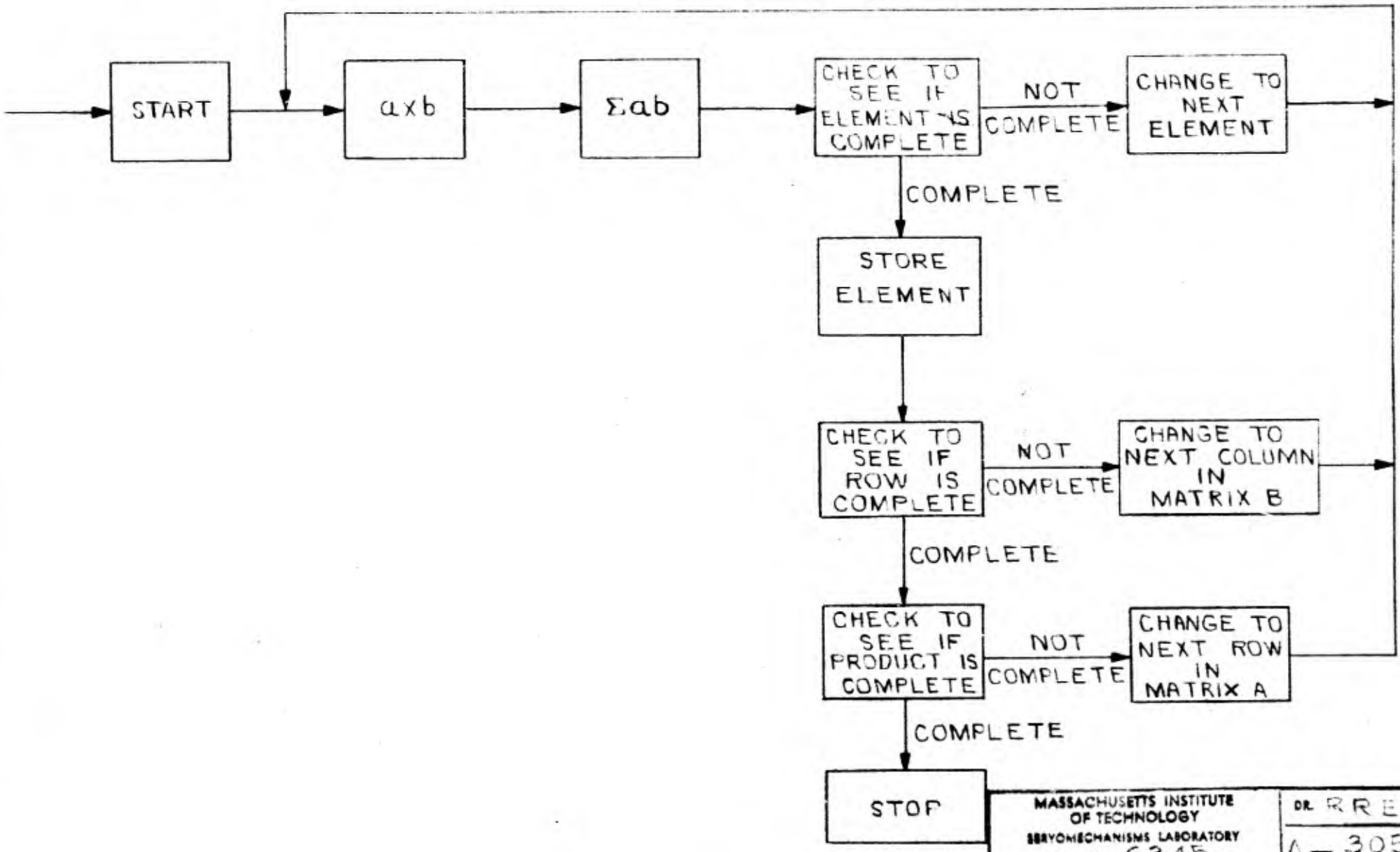
" " " 235 = 11101011

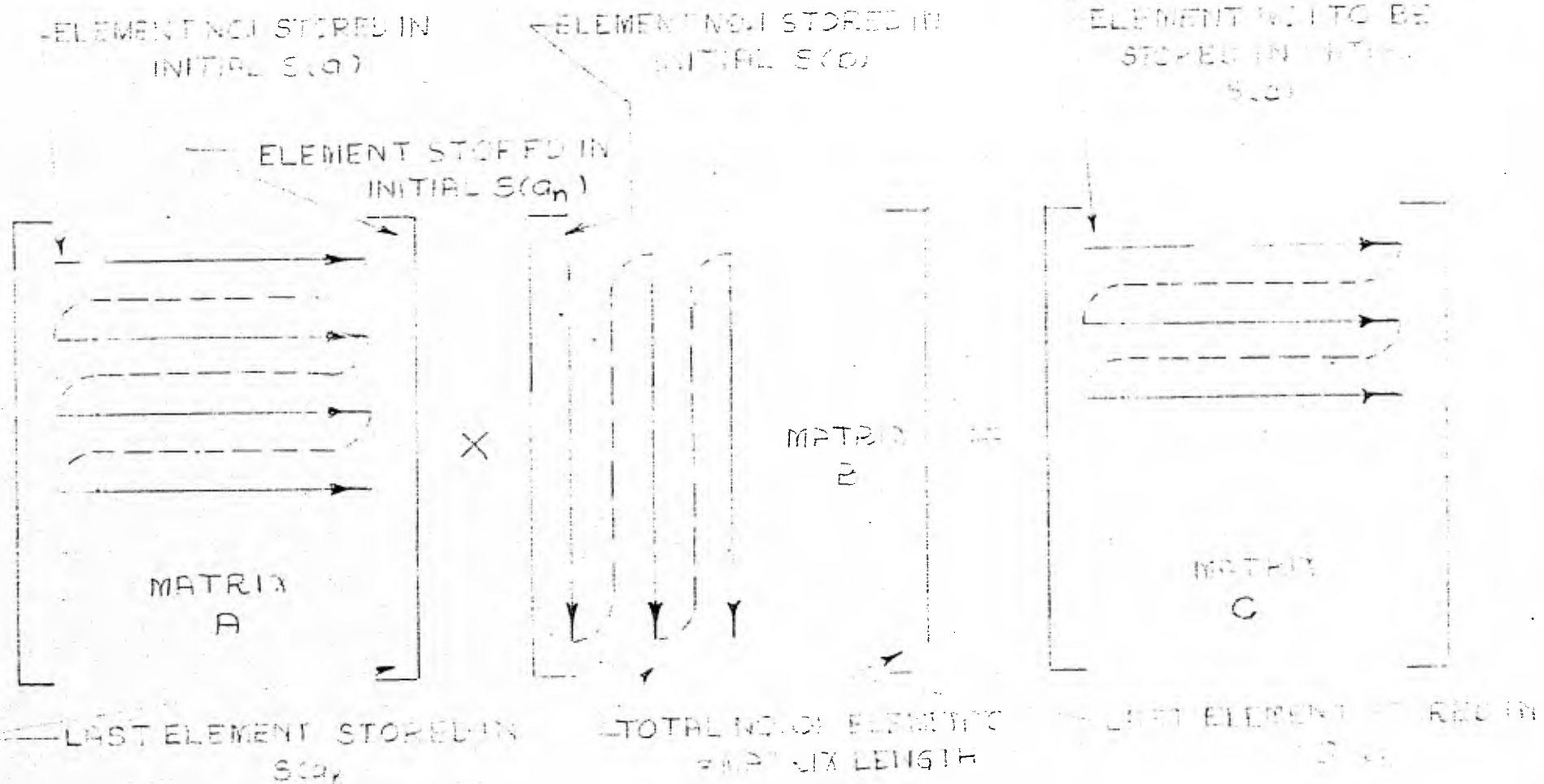
DECIMAL NUMBER 235

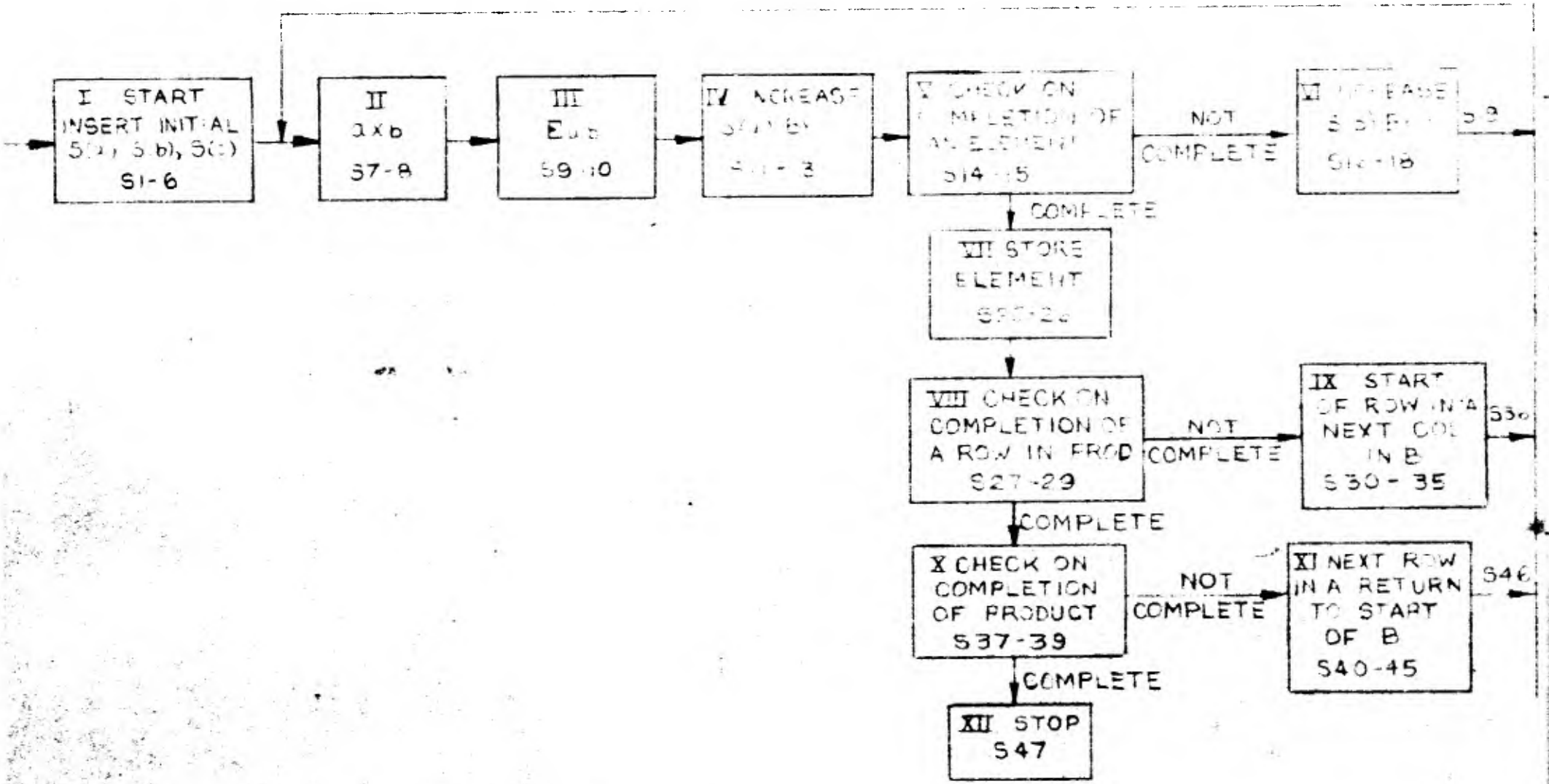
235 ÷ 2 = 117 + 1
117 ÷ 2 = 58 + 1
58 ÷ 2 = 29 + 0
29 ÷ 2 = 14 + 1
14 ÷ 2 = 7 + 0
7 ÷ 2 = 3 + 1
3 ÷ 2 = 1 + 1
1 ÷ 2 = 0 + 1

RESULT = 11101011

A-30418


START
axb
Σab
CHECK TO SEE IF ELEMENT IS COMPLETE
NOT COMPLETE
CHANGE TO NEXT ELEMENT
COMPLETE
STORE ELEMENT
CHECK TO SEE IF ROW IS COMPLETE
NOT COMPLETE
CHANGE TO NEXT COLUMN IN MATRIX B
COMPLETE
CHECK TO SEE IF PRODUCT IS COMPLETE
NOT COMPLETE
CHANGE TO NEXT ROW IN MATRIX A
COMPLETE
STOP
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
SERVOMECHANISMS LABORATORY
D.I.C. NO. 6345
DR. R R E
A-30501




I START
INSERT INITIAL
S(a), S(b), S(c)
S1-6
II
a x b
S7-8
III
E = a
S9-10
IV INCREASE
S11-13
V CHECK ON
COMPLETION OF
AN ELEMENT
S14-15
NOT
COMPLETE
VI INCREASE
S16-18
COMPLETE
VII STORE
ELEMENT
S19-26
VIII CHECK ON
COMPLETION OF
A ROW IN PROD
S27-29
NOT
COMPLETE
IX START
OF ROW IN A
NEXT COL
IN B
S30-35
COMPLETE
X CHECK ON
COMPLETION
OF PRODUCT
S37-39
NOT
COMPLETE
XI NEXT ROW
IN A RETURN
TO START
OF B
S40-45
COMPLETE
XII STOP
S47
MASSACHUSETTS INSTITUTE
OF TECHNOLOGY
SERVOMECHANISMS LABORATORY
DR.

MEMORANDUM NO M-6[illegible].

WRITTEN BY: David R. Brown

SUBJECT: High-Speed Digital-Computer Circuits

DATE: April 2, 1947

0345
Page 1 of 11

Illustrations
A-30339 A-30437
A-30429 A-30436
A-30401 A-30468
A-30438 A-30469
A-30426 A-30424
A-30431 A-30470
A-30427-1 A-30435
A-30433-1 A-30471
A-30432-1

Introduction

The circuits to be discussed are those of the Whirlwind I computer, Whirlwind I being a parallel-type, high-speed, digital computer, at present in the design stage. A presentation of all the circuits of the computer would be, of course, impossible in the time available. Only a small portion of the computer will be discussed, those parts of the arithmetic element used for the addition of two binary numbers. Limiting the discussion to a small part of the computer will make possible a more logical and detailed treatment of that part than would be possible if an attempt were made to discuss the entire computer. From the circuit point of view, very little is lost by thus limiting the discussion. The various parts of the computer are actually built up from a few basic circuits envolving only one or two tubes. These basic circuits may be regarded as the building blocks for the computer. The difference between the part of the computer to be discussed and the other parts, is in the arrangement of these basic circuits. The arrangement of the basic circuits is best shown by the block diagrams. Block diagrams are to be discussed in next week's paper.

First, the circuits in the arithmetic element necessary for the addition of two binary numbers will be presented. Second, some general principles of circuit design will be discussed. Third, some results of the experimental work done by the laboratory will be presented.

Circuits for Addition

Before the discussion becomes confined to the arithmetic element, the relationship of the arithmetic element to the rest of the computer must be shown.

The five parts of the computer are shown in Illustration A-30339. The control sends to the other parts of the computer the timing pulses which

are necessary to perform the operations called for by the program. The storage provides storage for the program, input data, and partial results. The arithmetic element can add, subtract, multiply, or divide two numbers as dictated by the control. The input and output are also shown.

The parts of the arithmetic element which are used in addition, with the connections from the control, are shown in Illustration A-10429. Two registers are used, the A register and the accumulator. A register has a digit-storage element in each column and the accumulator is a register which can add two binary numbers. The A register and the accumulator will be discussed in more detail later.

Now consider the steps which are necessary to execute the addition of two numbers with storage of the sum. Initially, both the A register and the accumulator are clear. The augend will appear on the digit-transfer bus at some known time. The digits of the augend will be represented by short voltage pulses and will appear simultaneously on the bus cables, there being one bus cable for each digit column. The presence of a pulse will represent a one and the absence of a pulse will represent a zero. At the correct time, by means of the connection labeled bus to A register, the control will open the gate and let the augend go from the bus to the A register. Next the gate between the A register and the accumulator is opened by means of the connection labeled A register to accumulator, permitting the number to go to the accumulator. Then the A register is cleared by means of the connection labeled clear A register. At the proper time, the first gate is opened again and the addend is permitted to go from the bus to the A register. Then the gate between the A register and the accumulator is opened and the addend goes to the accumulator where it is added to the augend. To complete the addition, the connection from the control labeled carry is used. The carry will be described later. By means of the connection labeled accumulator to bus, the sum may be sent to the digit-transfer bus.

The functions necessary to perform the addition of two binary numbers are the functions necessary in the accumulator. So that these functions may be clearly understood, binary addition will be reviewed. The four possible cases of addition of two binary digits are shown:

0	1	0	1 ← augend
+0	+0	+1	+1 ← addend
0	1	1	1 ← carry
			0 ← partial sum
			10 ← sum

The procedure for binary addition with ordinary single-carry steps is as follows: add the two numbers and obtain a partial sum and carries. Shift the carries one column to the left and add to the partial sum, obtaining a partial sum and carries. Shift the carries again and add, etc., until no carries remain. When dealing with N-digit numbers, a carry may have to be shifted N times. An example of this is shown.

```
 0001 <-- augend
 1111 <-- addend
   1  <-- 1st carry
 1110 <-- partial sum
  1
   1  <-- 2nd carry
 1100 <-- partial sum
 1
   1  <-- 3rd carry
 1000 <-- partial sum
1
  1   <-- 4th carry
 0000 <-- partial sum
1
10000 <-- sum
```

This is the equivalent of adding one to 9,999 in the decimal system. After some practice, one learns to write the final sum in one step instead of five as in the example. The accumulator can be constructed to perform an addition in two steps instead of N+1 steps. The two steps are: one, obtain partial sum and carries; two, propogate carries to the left and add. A binary addition is shown in Illustration A-30401. Note that a carry cannot propogate to the left past a zero in the partial sum.

The basic circuit used as the storage element in the registers is a flip-flop (resistance-coupled multivibrator or Eccles-Jordan trigger circuit). A circuit diagram of a flip-flop is shown in Illustration A-30428. The flip-flop has two stable states: one, V_1 conducting and V_2 cutoff; two, V_1 cutoff and V_2 conducting. Negative pulses are used to set, reset, or trigger the flip-flop. When reset by means of a negative pulse at the reset terminal, the digit stored is a zero and the plate of V_1 is at a low potential. The fact that V_1 is shaded, indicates that V_1 is normally conducting or conducting when reset. When set by means of a negative pulse at the set terminal, the digit stored is a one and the plate of V_1 is at a high potential. The digit stored may be determined by sensing the voltage at the plate of V_1. When triggered by means of a negative pulse at the trig. terminal, the flip-flop will transfer to the opposite state regardless of its state when triggered. The symbol for a flip-flop is also shown in the Illustration.

Another basic circuit is the gate circuit. A gate circuit is shown in Illustration A-30425. Grid 1 or grid 2 can cutoff plate current; i.e., plate current can flow only when grid 1 and grid 2 are above cutoff. Both grids are normally biased below cutoff. The input and gate pulses must be positive and a negative output pulse is obtained. The symbol for a gate tube or gate circuit is also shown.

A gate circuit which does not invert the input pulse is shown in Illustration A-30431. A pulse transformer is used in the plate circuit to reinvert the pulse. The pulse transformer will be described in some detail later. The same symbol is used for this gate circuit as is used for the first gate circuit.

The A register may be constructed from these two basic circuits, the flip-flop and the gate circuit. Two digit columns, the first digit column and the second digit column, of the A register are shown in Illustration A-30427. A gate pulse from the control, by means of the connection labeled bus to A register, lets the number on the bus cables go through the read-in gate tubes and set the flip-flops. Thus the number is transferred from the bus to the A register. A pulse from the control on the connection labeled A register to accumulator will pass through the read-out gate tubes held open by the flip-flops and go to the accumulator. A pulse from the control on the connection labeled clear will clear the A register by resetting all the flip-flops which have been set.

An accumulator using ordinary single-carry steps for addition is shown in Illustration A-30433. Only two digit columns are shown. Two flip-flops are used in each column; one set to store the partial sum, the partial sum flip-flops; and the other set to store the carries, the carry flip-flops. Suppose the augend is already in the accumulator and the addend is about to be added to it. The carry flip-flops are all zero. The digit pulses from the A register go to the trigger terminals of the partial-sum flip-flops. In all columns where a one is added to a one, the digit stored in the partial-sum flip-flop becomes a zero and the digit stored in the carry flip-flop becomes a one. One plate terminal of the partial-sum flip-flop is coupled to the set terminal of the carry flip-flop so that whenever the partial-sum flip-flop is triggered from one to zero, the carry flip-flop is set. In any column there are only three possibilities: one, both partial-sum and carry flip-flop zero; two, partial-sum flip-flop one and carry flip-flop zero; three, partial-sum flip-flop zero and carry flip-flop one. Both the partial-sum and carry flip-flop cannot simultaneously contain a one.

The next steps of the adding sequence are to shift and add the carries as many times as are necessary. Transfer the carries one column to the left and add. A pulse from the control on the carry connection will

go through the carry gate tubes held open by the carry flip-flops which store ones. If a carry flip-flop stores a one, the pulse will pass through an open carry gate tube and trigger the partial-sum flip-flop in the next column. The carry flip-flop is reset by the carry pulse a short time later. The delay elements used are simply short pieces of cable having a relatively low velocity of propagation. If the partial-sum flip-flop in the second column already has a one in it, a carry digit stored in the carry flip-flop of that column must result. In this event, the carry flip-flop should not be set until after the delayed carry pulse from the control has reset the carry flip-flops.

The carry digits may have to be transferred and added N times. This means the control may have to send carry pulses N times.

An accumulator for high-speed addition is shown in Illustration A-30432. Only two digit columns are shown. Again suppose the augend is already in the accumulator and the addend is about to be added to it. If the partial-sum flip-flop in a given column already has a one stored in it, a digit pulse from the A register will go through the store-carry gate tube held open by the partial-sum flip-flop and set the carry flip-flop, the partial-sum flip-flop being triggered back to zero a short time thereafter. If the partial-sum flip-flop had stored a zero and a one was added, the carry flip-flop would not have been set.

This accumulator uses high-speed carry. A single pulse from the control on the carry connection will completely clear the carry flip-flops and will cause the sum to appear in the partial-sum flip-flops. However, a carry digit may still have to be propagated the length of the register. In the high-speed carry, one carry digit may travel the length of the register.

The high-speed carry circuit operates as follows: a pulse from the control on the carry connection finds, say, the carry gate tube of the first digit column open, i.e., a carry digit stored in the first digit column. The pulse, now representing the carry digit, goes to the high-speed-carry gate tube controlled by the partial-sum flip-flop in the second digit column. If that flip-flop stores a one, the carry digit will go on to the next column, etc. As it goes, it is delayed at each column and used to trigger the partial sum flip-flop. The carry flip-flop is also reset by the delayed carry pulse. A set of read-out gate tubes is controlled by the partial-sum flip-flops for sending the sum to the bus. The control can transmit the sum to the bus by means of the accumulator to bus connection. The control can clear the partial-sum flip-flops by means of the clear connection. The carry flip-flops are automatically cleared during the carry.

Principles of Circuit Design

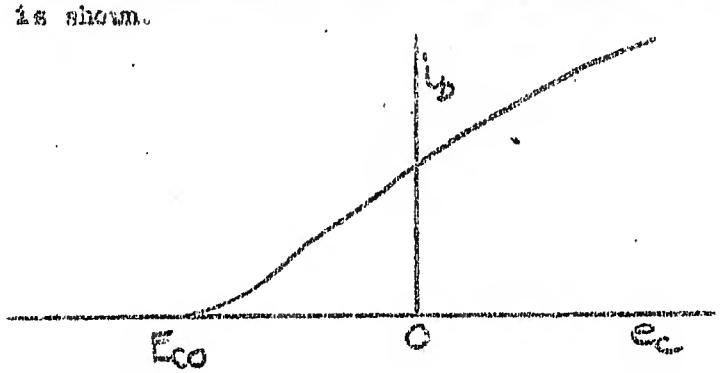
The general principles of circuit design may be summarized in one paragraph.

The problem must be solved in some given real time. A program of the solution of the problem will show the number of operations, i.e., transfers, additions, multiplications, etc., to be performed in the solution of the problem. The number of pulse intervals required for each operation will then determine the pulse repetition frequency. When the pulse repetition frequency is known, the order of magnitude of the pulse rise time and fall time is known. Now, vacuum tubes which are available in fairly large quantities and whose characteristics, especially reliability, are well established, should be used if possible. Conventional types of tubes should be used. Then, something is known of the physical size of the circuits and the approximate value of the shunt capacitances in the circuits can be estimated. Knowing the required rise time and fall time and the shunt capacitances, the load resistors are determined. When a tube type is selected, the approximate pulse amplitude required is determined by the transfer characteristic of the type selected. The problem is to pick a tube that will provide a current through the load resistor sufficient to produce the required pulse amplitude. Furthermore, by far the best gate tube at the present time is the 6AS6. The pulse amplitude for the Whirlwind I computer has been selected on the basis of the transfer characteristic of the 6AS6.

All the computer circuits can be constructed with flip-flops, gate tubes, crystal rectifiers, and delay lines as the building blocks. Also, buffer, or power, amplifiers, inverters, and pulse reshapers will be necessary.

Reliability is of utmost importance. A failure at one point will lead to a meaningless result. Components of high quality must be used with conservative ratings. Because large numbers of vacuum tubes will be used, approximately three thousand for Whirlwind I, vacuum-tube life is particularly important. The tube-life problem is a serious one. In an effort to prolong tube life, manufacturers' ratings of electrode dissipation will be reduced by a factor of two.

As was stated previously, the digit pulse amplitude will be determined by the transfer characteristic of the tubes used. A typical transfer characteristic is shown.



The normal operating point should be as far as possible below cutoff, E_{co}, to avoid trouble from spurious signals. However, operation far below cutoff will introduce more delay in a pulse having a finite rise time. Operation far below cutoff also requires a larger voltage change in the preceding stage, necessitating a larger tube in the preceding stage to obtain the same rise time.

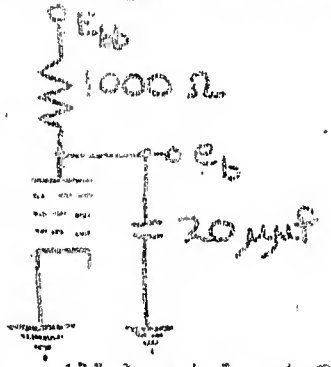
If the grid of the tube is not driven positive there is no possibility of grid clipping and the operating point for the top of the pulse will be on the steep portion of the transfer characteristic.

If the grid is driven positive and a clipping resistor is used, the rise and fall times introduced will be appreciable. Consider the 6AS6 gate circuit shown in Illustration A-30437. The d-c transfer characteristic is also shown. Make E_{col} equal to 10 volts. Drive the grid to +2 volts; at this point 0.4 milliamperes of grid current flow and the grid has a static resistance of five thousand ohms. Suppose a twenty-thousand-ohm limiting resistor, R_g, is used so that the input terminal must be driven to +10 volts. The input capacitance, C_{in}, of the tube and socket might be as low as 5 micromicrofarads. If a 20-volt, 0.25 microsecond pulse having vertical rise and fall is applied to the input terminals of the circuit, the resulting waveform at the grid will be similar to that shown in Illustration A-30436. The time constant of the rise and fall in the negative grid region is 0.1 microsecond. Note the delay and lengthening introduced.

As a third alternative, the grid may be driven positive with no grid limiting resistor. The operating point will then be on the steep part of the transfer characteristic. The maximum safe grid voltage is established by the power the electrodes within the tube can dissipate and the current that can be drawn from the cathode.

In the Whirlwind I computer, two pulse repetition frequencies will be used. One-megacycle pulse repetition frequency will be used for everything except the arithmetic element and a four-megacycle pulse repetition frequency will be used for the arithmetic element. This results in pulse intervals of one microsecond and 0.25 microsecond. The pulse length must be short compared to the pulse interval because a pulse will be lengthened and delayed in going through gate tubes, power amplifiers, etc. A pulse representing a digit must not slop over into a time interval belonging to some other number or order. As a consequence, the fall time is as important as the rise time. Rectangular voltage pulses are preferred because a rectangular pulse can go through these extremely non-linear gate tubes, etc., and come out as a rectangular pulse with no delay with respect to the input pulse. Of course rectangular pulses cannot be realized. A pulse as nearly rectangular as practical will be used and reshaped when necessary.

In order to achieve short rise and fall times, the load resistors must be kept small. Shunt capacitance is encountered everywhere. Consider the plate load of a gate circuit using a 1,000-ohm plate load resistor. The circuit diagram is shown.

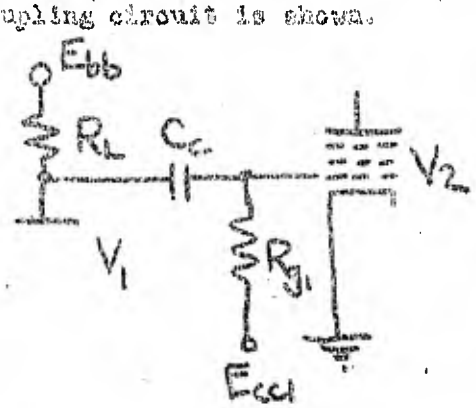


The plate to ground capacitance will be at least 20 micromicrofarads. The fall of the output pulse will have a time constant of 0.02 microseconds; the rise time will be shorter than the fall time because the load resistor is effectively paralleled by the plate resistance of the tube during the rise. The waveform at the plate will be similar to the waveform shown.

A small inductance in series with the plate load resistor will improve the rise and fall.

Some overshoot will be introduced as shown.

The difficulty in coupling is that the time interval between pulses is often variable. The interval may vary between 0.25 microsecond and ten minutes. A typical coupling circuit is shown.



is waveform D. The waveform of the delayed pulse after V_3 in waveform E. Only two stages of the high-speed-carry circuit were constructed for these tests.

The circuit diagram of a high-speed flip-flop is shown in Illustration A-30425. The flip-flop uses pentodes, 6AG7's, to reduce the grid-to-plate capacitance. Bias for the grid return is obtained by use of a 600-ohm cathode resistor; the current through this resistor is constant. Note the use of the 10-micromicrofarad compensating capacitors to improve the rise time at the control grid. Also, note the low-resistance load and plate-to-grid voltage divider. Only the trigger-input terminal is shown. Waveforms at one plate terminal of the high-speed flip-flop are shown in Illustration A-30422. Waveform A is for a 0.875-megacycle trigger. The amplitude of the voltage change is 25 volts and the rise time and fall time are less than 0.1 microsecond. Waveform B is for a 7-megacycle trigger.

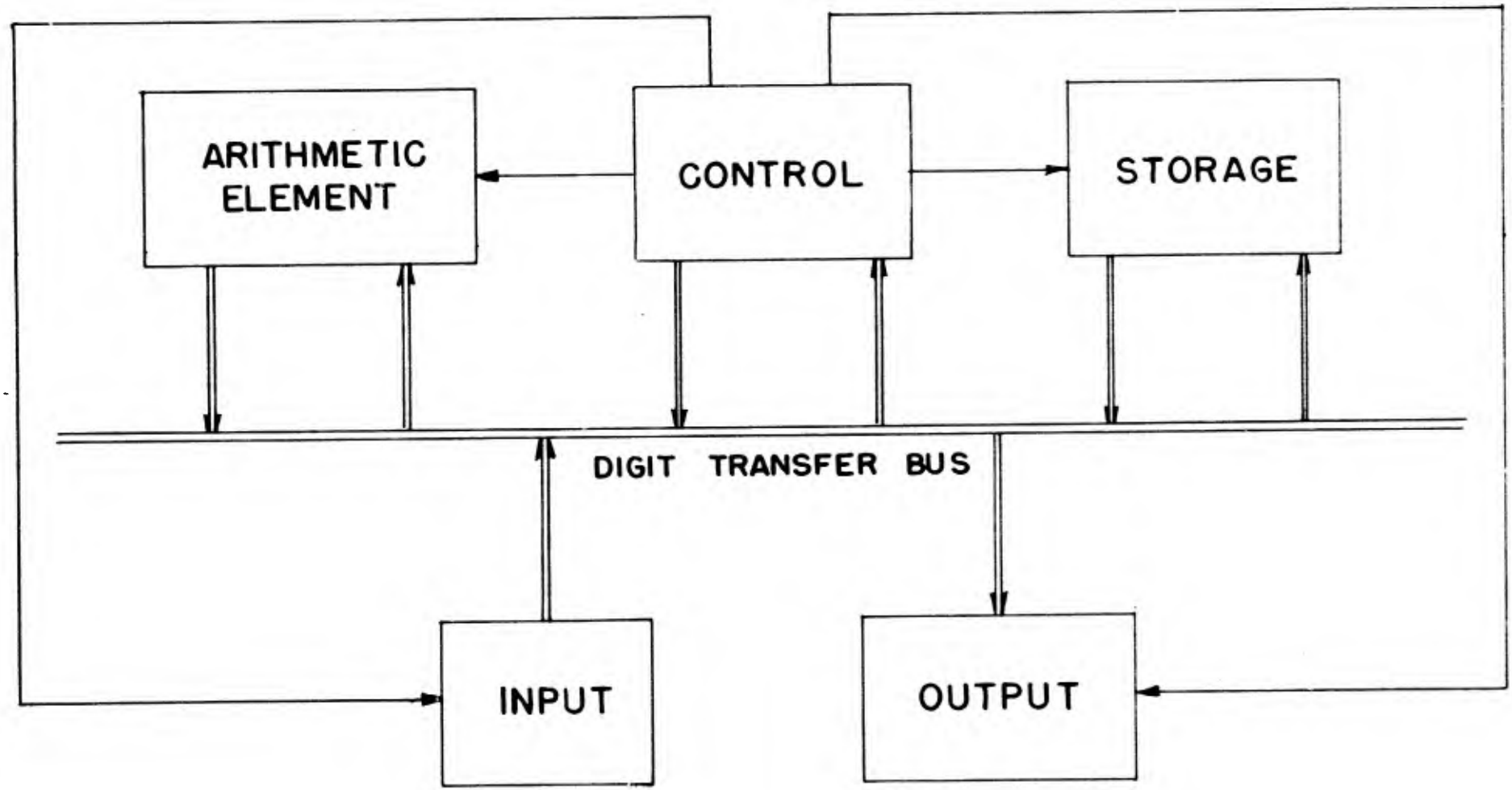
Conclusion

The outstanding problem faced at the present time is the tube-life problem. A high duty factor, as high as 0.25, aggravates the problem. The effect of electrode dissipation, cathode current, and heater voltage on tube life is not adequately understood. The variable pulse repetition frequency causes many coupling difficulties. In order to facilitate checking, some direct coupling will be used. Checking methods have received little attention at the present time.

David R. Brown

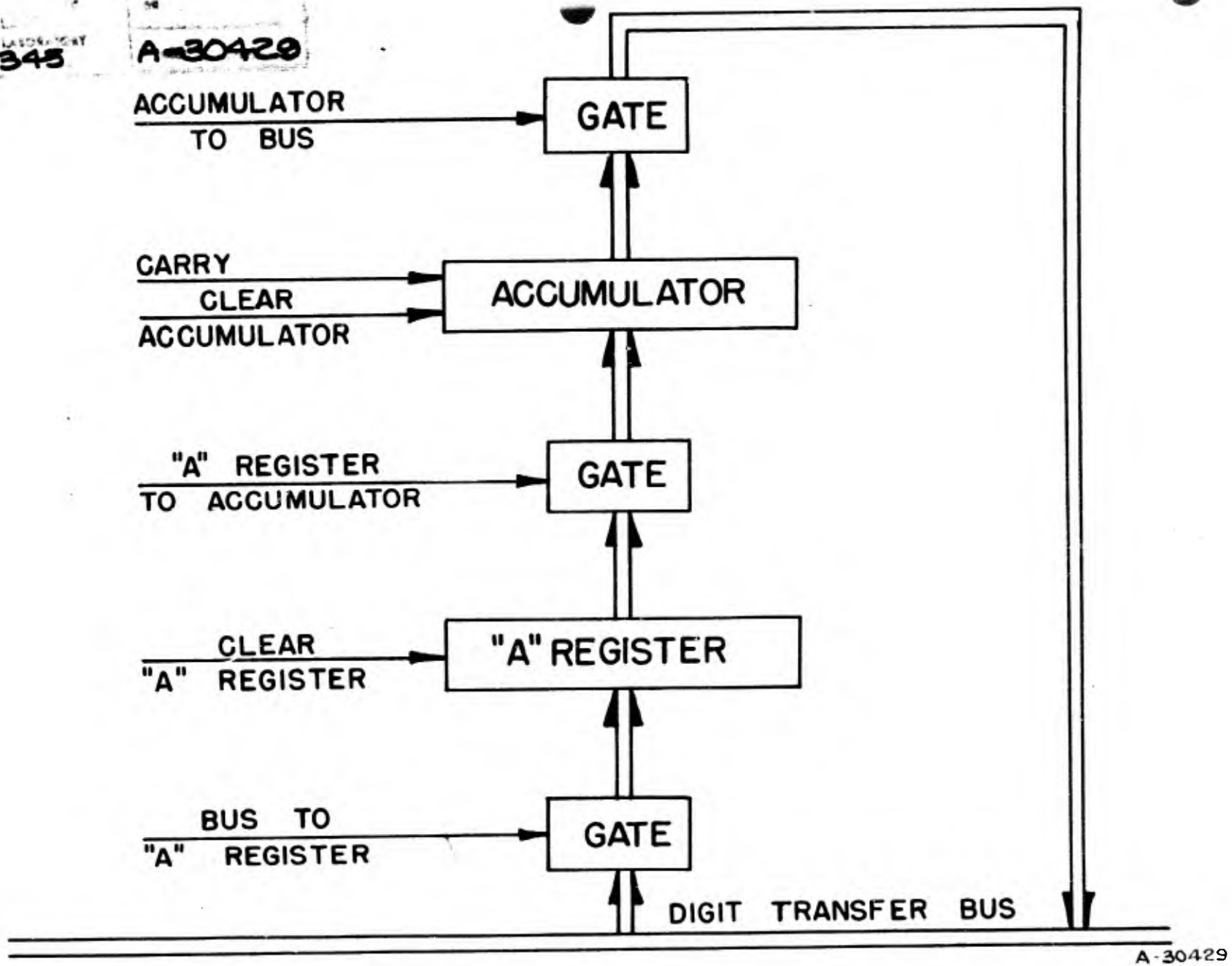
DR:vb

A-30339-1



6345 RLK.
RICE

A-30339-1



ARITHMETIC ELEMENT CONTROLS FOR ADDITION

6345
RRE
3/20/47

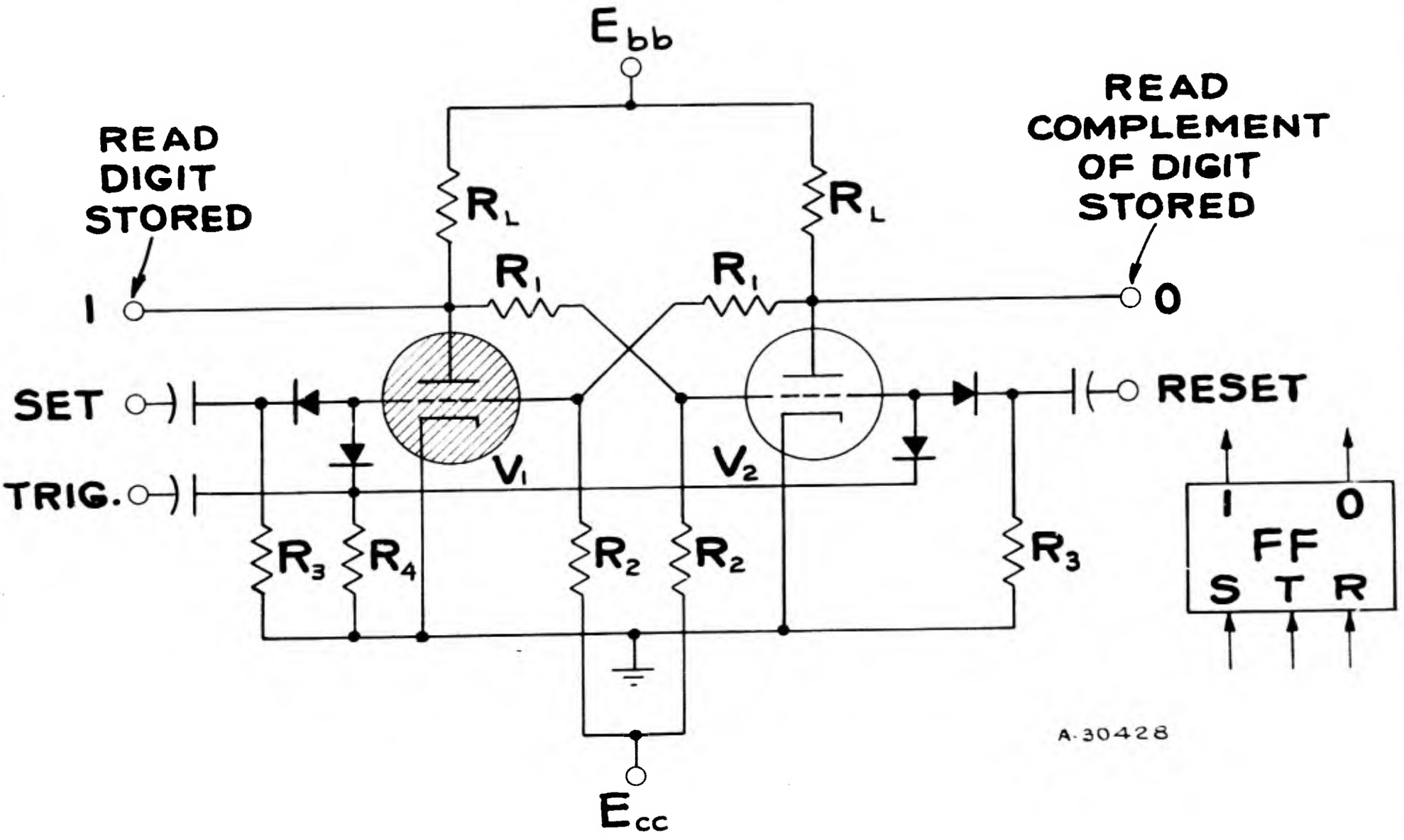
A-30401

BINARY ADDITION

```
  11011010
  10110110
 ----------
  101101100
 ----------
  1010010000
 ----------
  1100000000
 ----------
  110010000
```

A-30401

A-3042

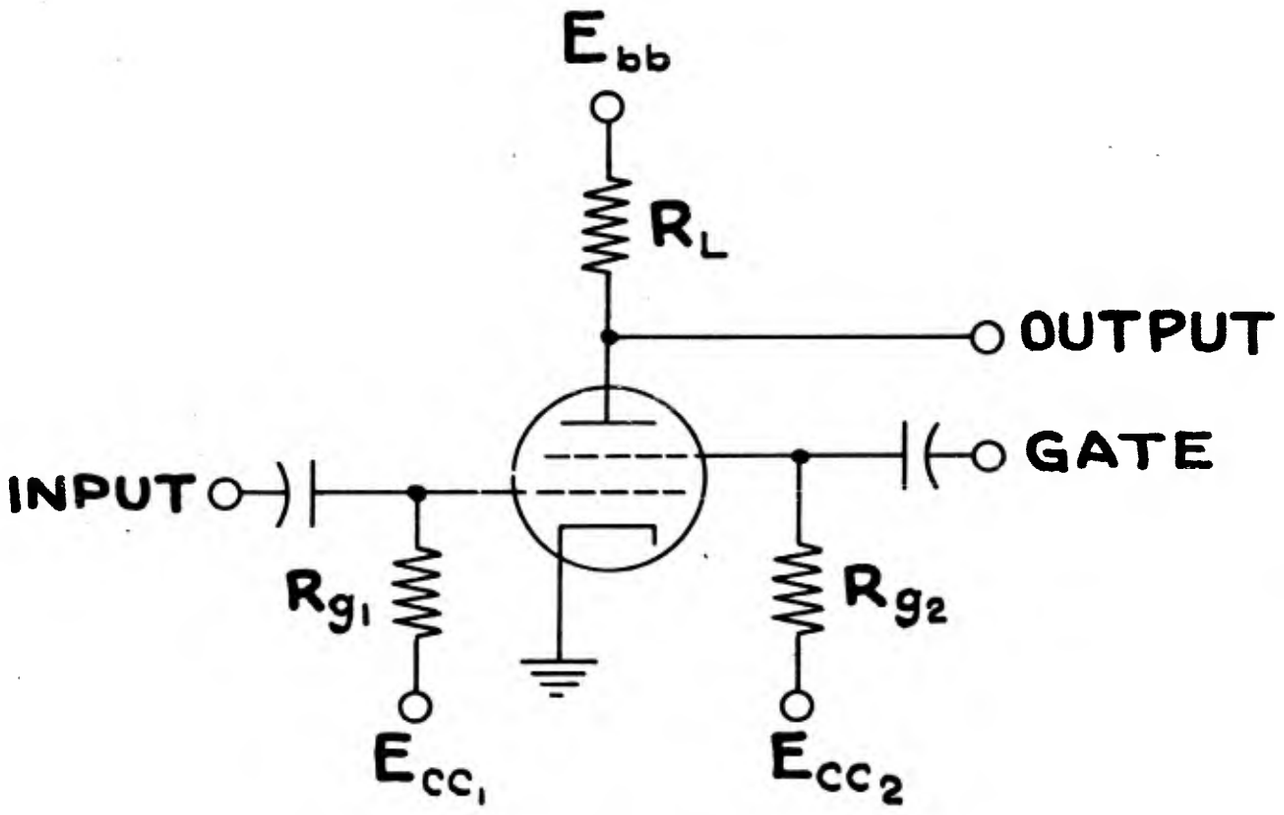


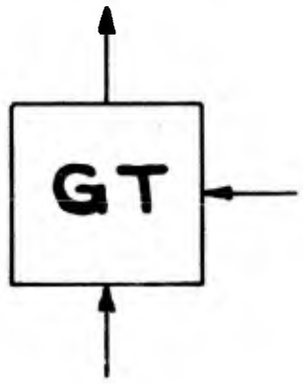
FLIP-FLOP

6345 A-30428

A-30426

GATE CIRCUIT

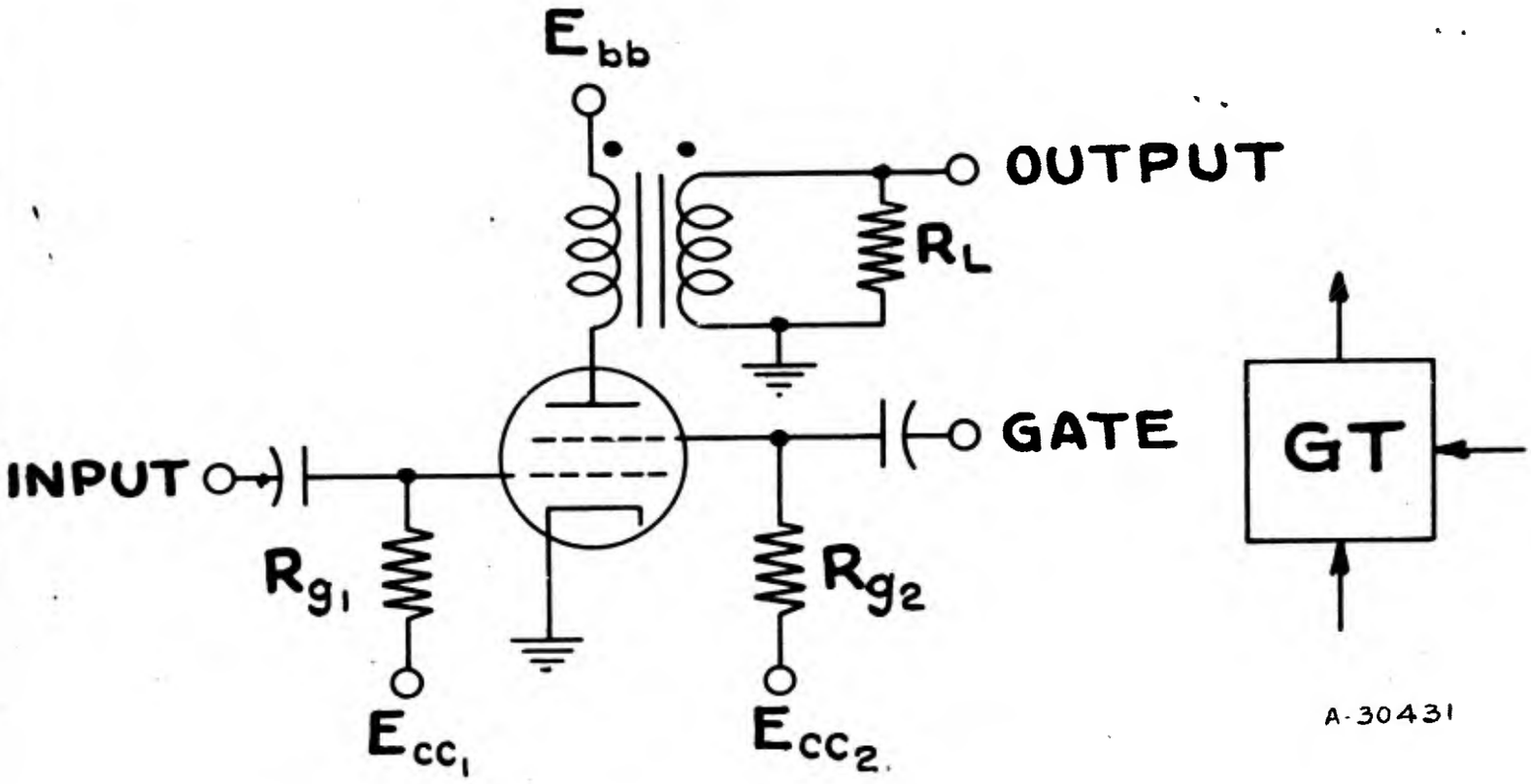




A-30426

A-30431

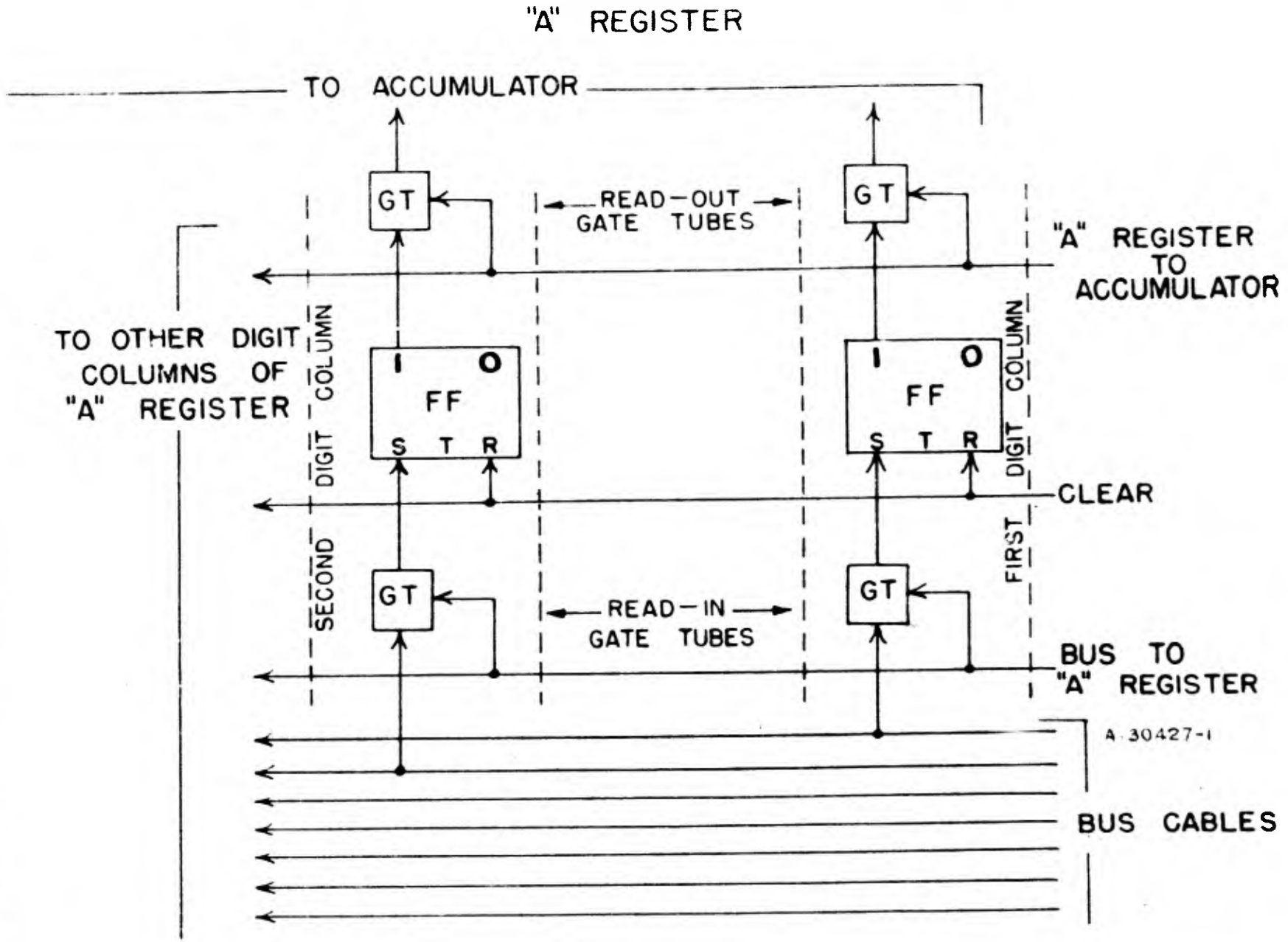
GATE CIRCUIT



A-30431

A-30431

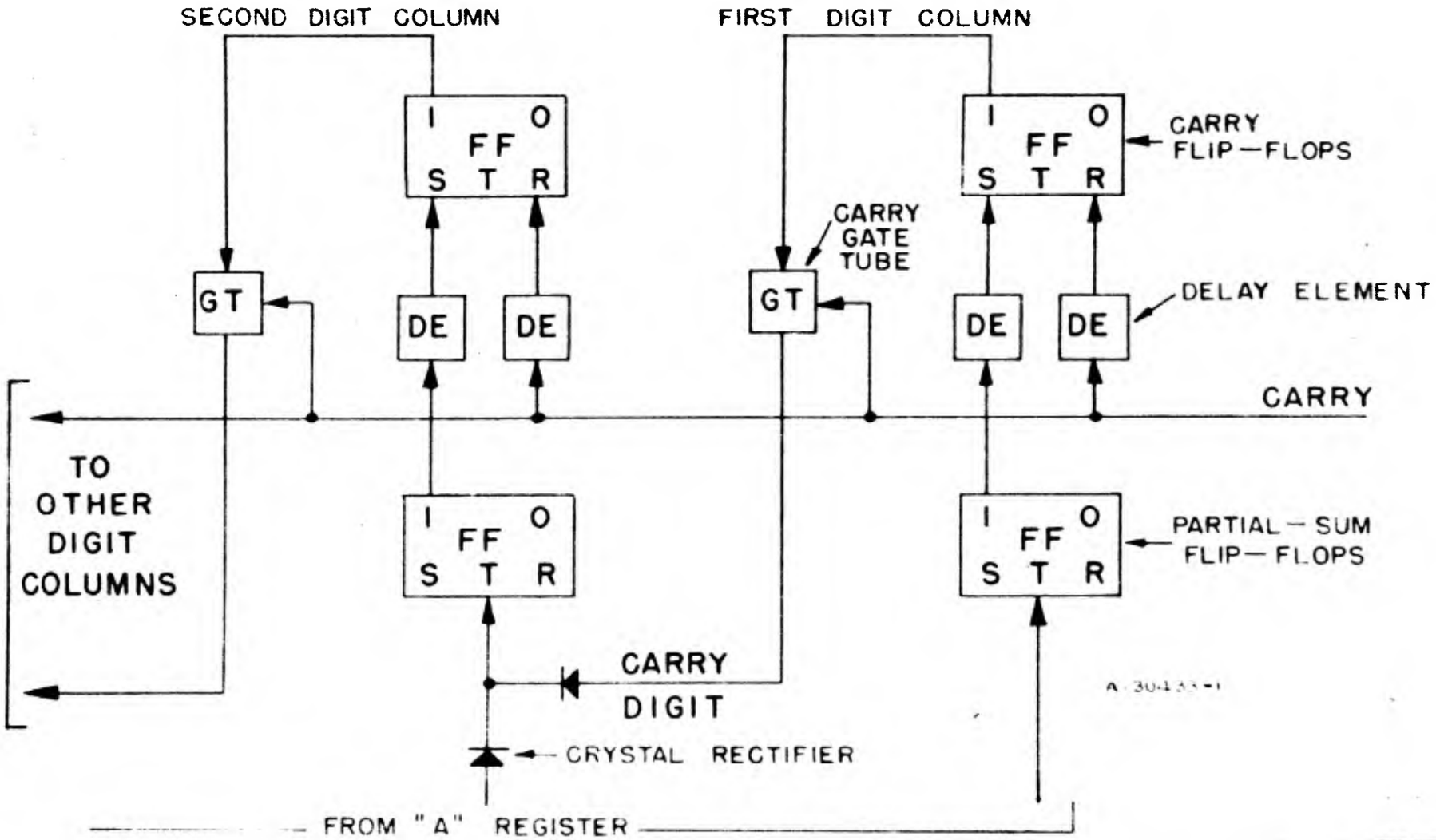
1-30427-1



A-30433-1

A-30433-1

ACCUMULATOR

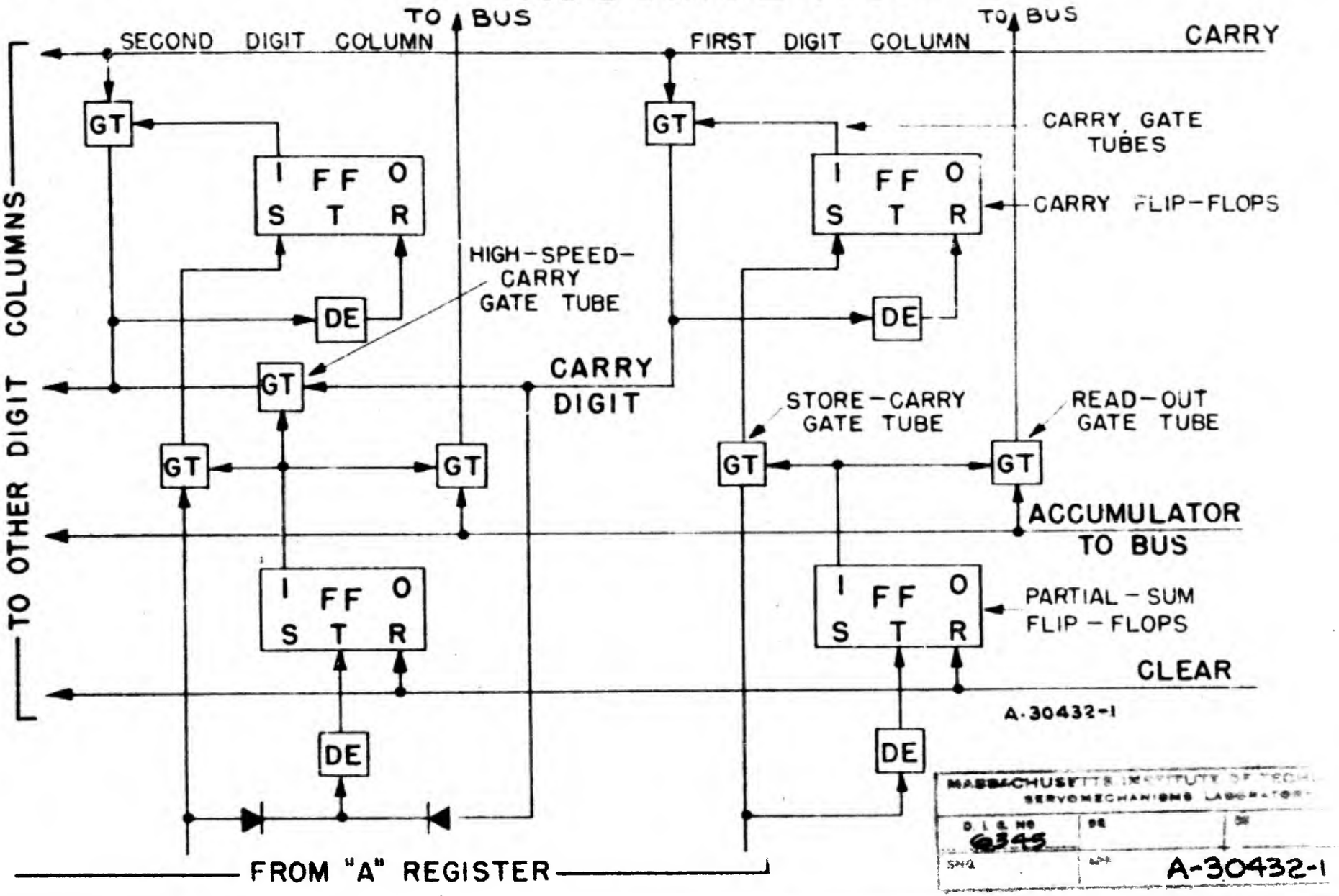


A-30433-1

1-30432-1

A-30432-1

ACCUMULATOR

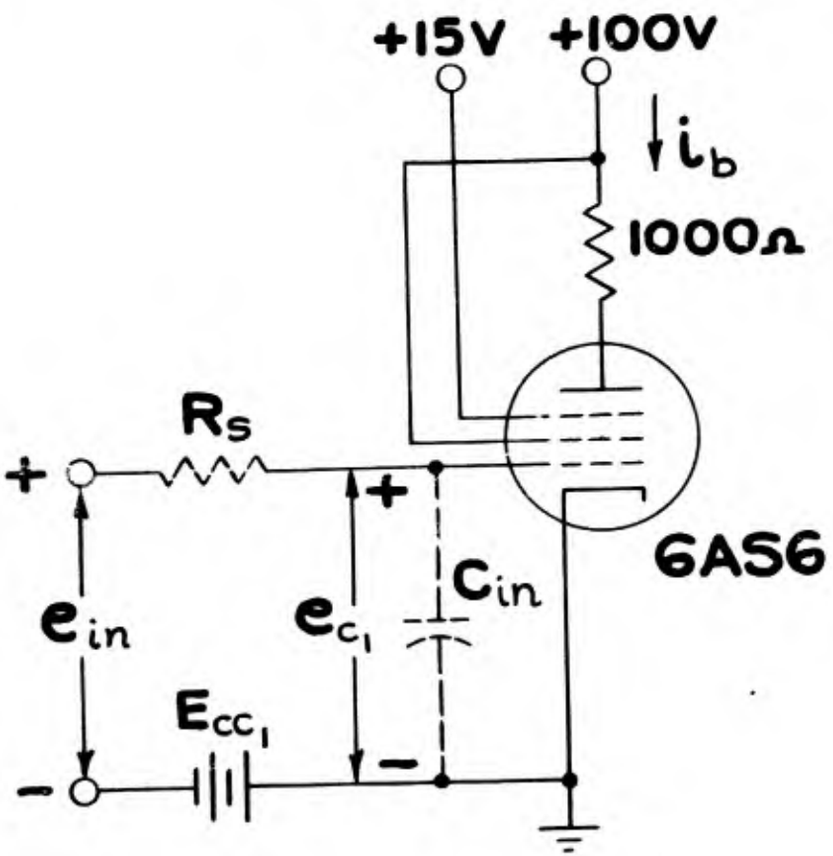


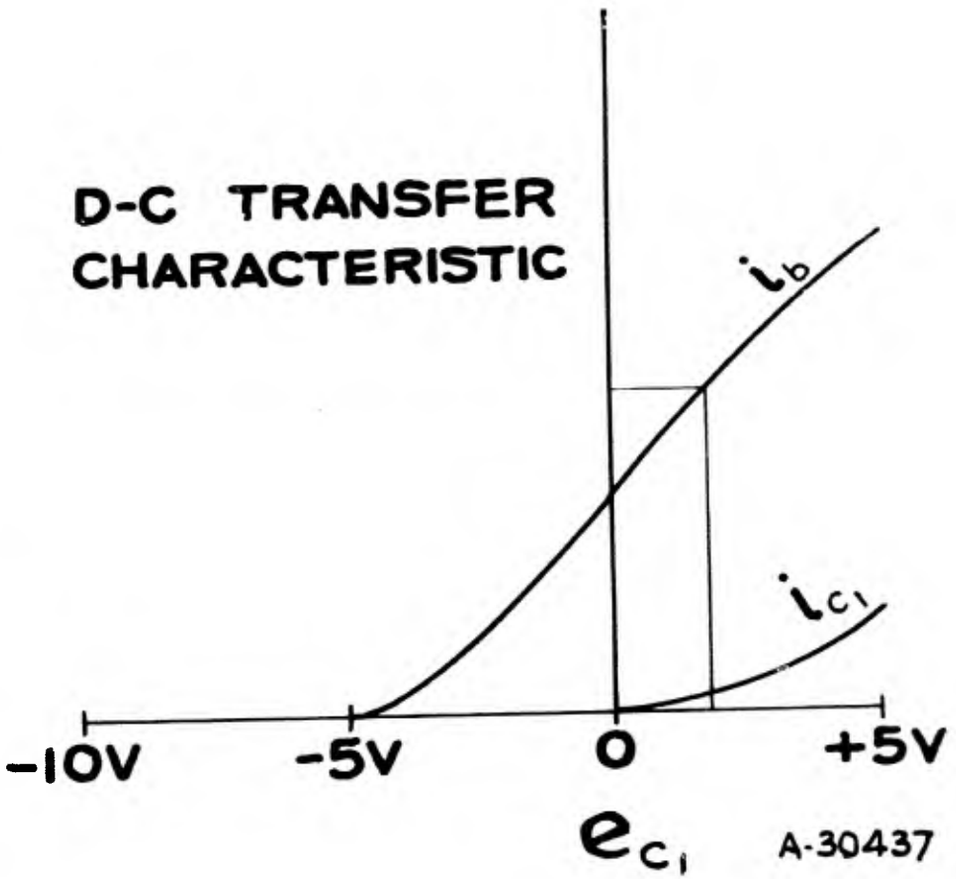
MASSACHUSETTS INSTITUTE OF TECHNOLOGY
SERVOMECHANISMS LABORATORY

D L NO 6345

A-30432-1

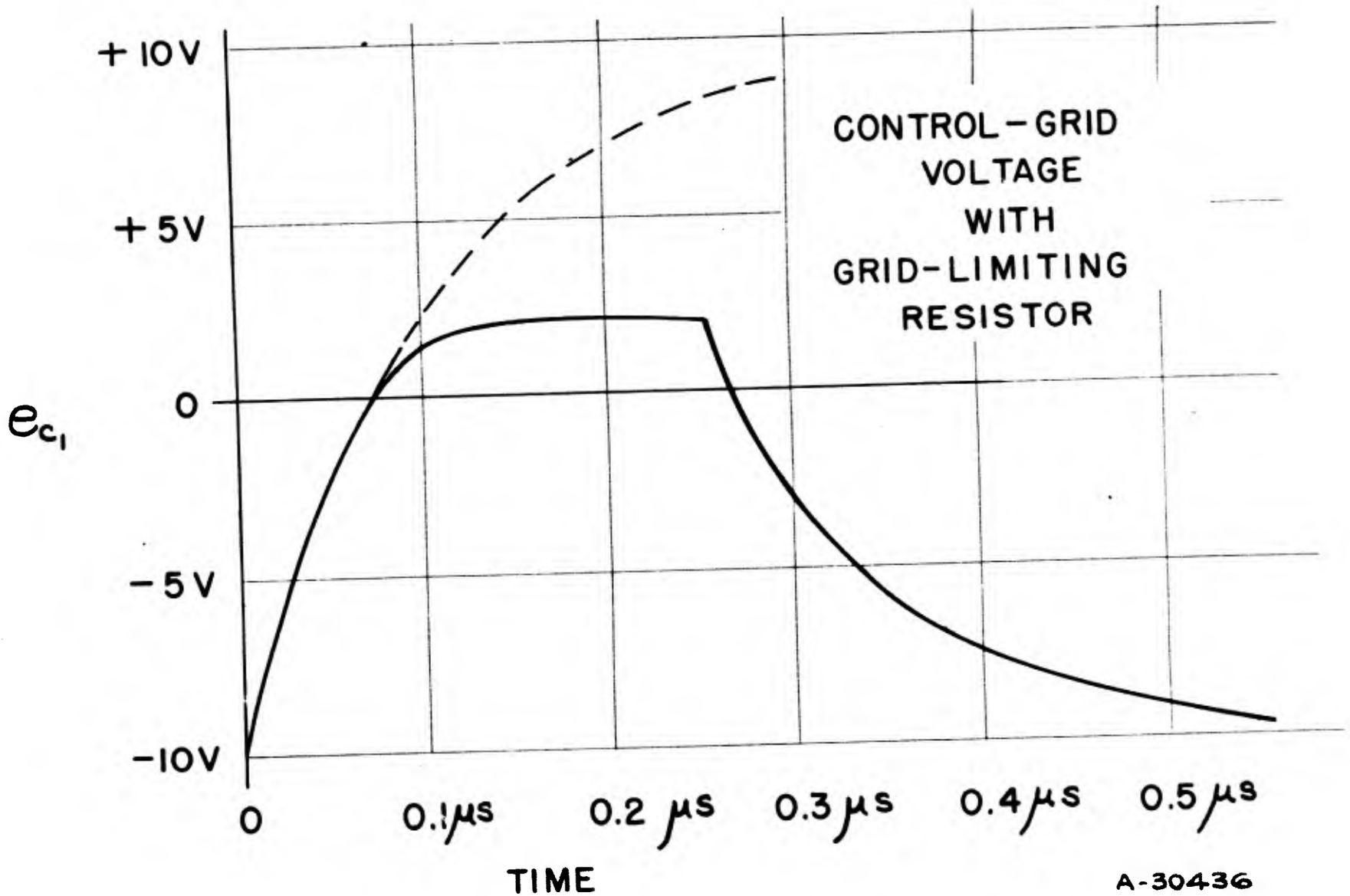
A-3043



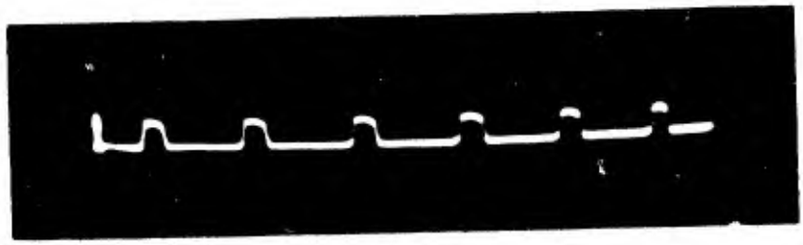


6345 A-30437

A-30436



A-30436



I-II

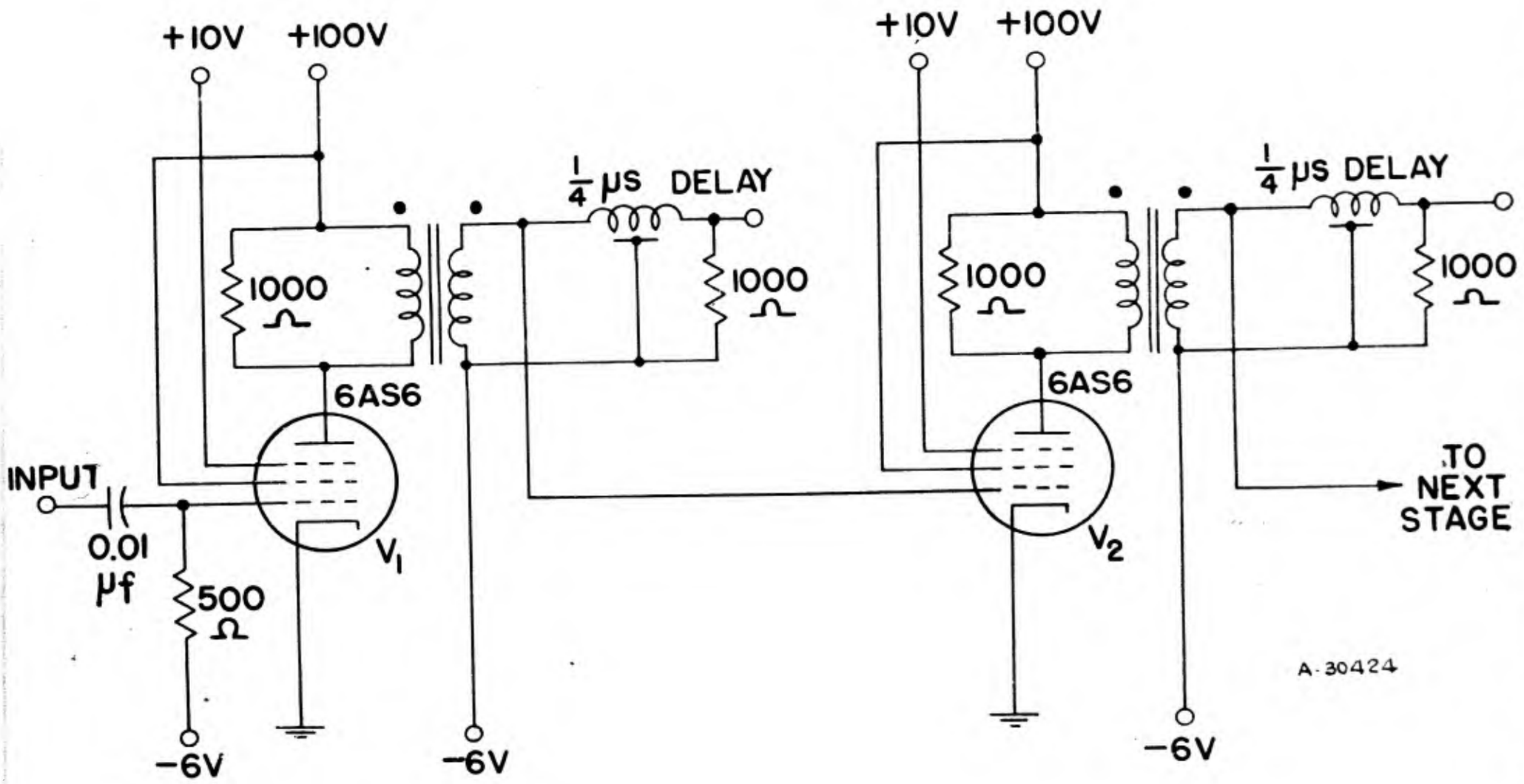
ONE-MEGACYCLE CLOCK PULSES

A-30468

1-6

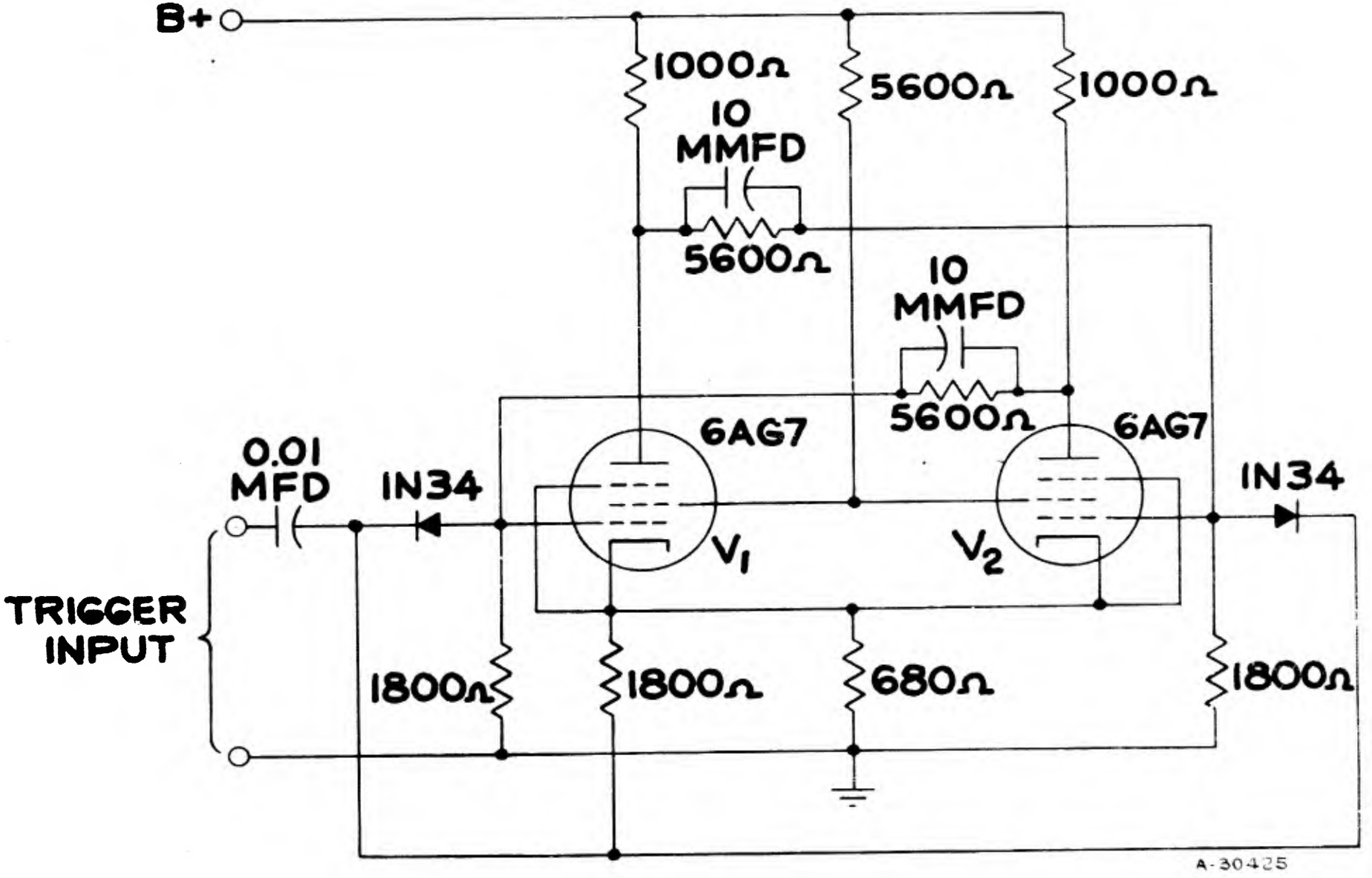
INPUT AND OUTPUT WAVEFORMS FOR ONE-TO-ONE INVERTER TRANSFORMER

A-30424

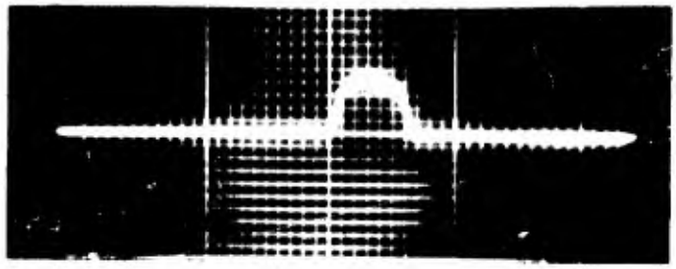


GATE TUBES IN SERIES

6345 A-30425



HIGH-SPEED FLIP-FLOP

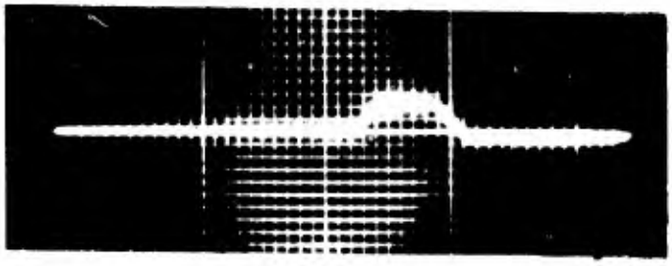


F29-4

A. INPUT PULSE

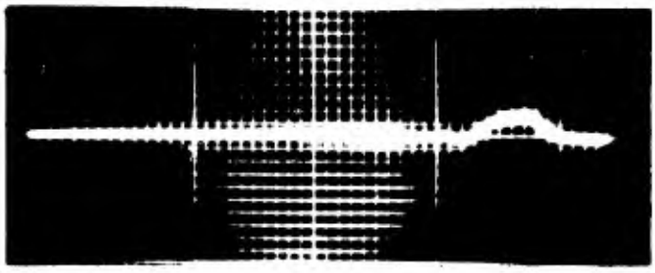
F29-5

B. PULSE AT GRID OF V2



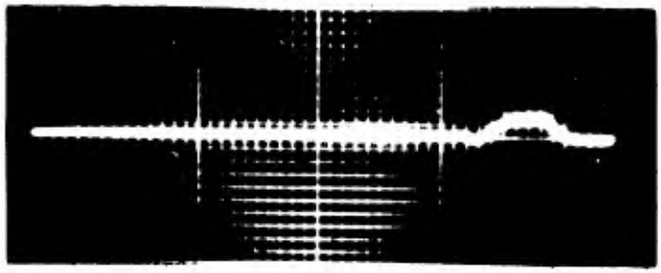
F29-6

C. PULSE AT GRID OF V3



F29-7

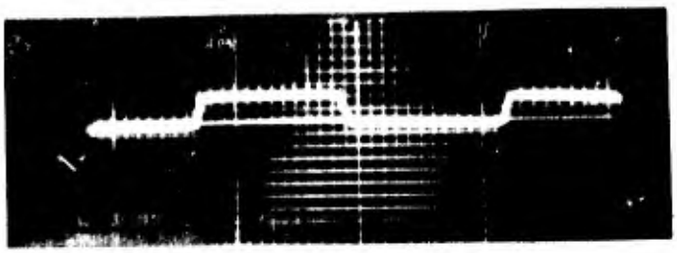
D. DELAYED PULSE AFTER V1



F29-8

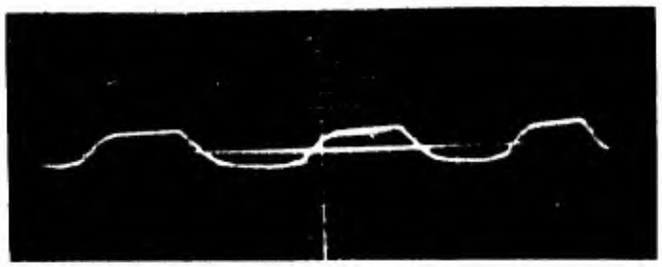
E. DELAYED PULSE AFTER V2

WAVEFORMS FOR HIGH-SPEED CARRY



F13—4

A. 0.875—MEGACYCLE TRIGGER



F13—3

B. 7—MEGACYCLE TRIGGER

WAVEFORMS AT ONE PLATE OF HIGH—SPEED FLIP—FLOP

A-30471

MEMORANDUM M-69

Servomechanisms Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

Written by: Robert R. Everett

6345

Subject: Digital Computer Block Diagrams
lecture given for the M.I.T. Electrical Engineering Staff Seminar

Date: April 9, 1947

A-30339 A-30454
A-30440 A-30452
A-30441 A-30404
A-30442 A-30447
A-30443 A-30448
A-30444 A-30449
A-30445 A-30450
A-30446 A-30451

We shall begin with the standard computer block diagram, Illustration A-30339, used in several of the previous seminars. This diagram shows the basic components of the computer. These components have been discussed briefly before but more from the point of view of what they can do rather than how they do it. Mention has been made of some of the arithmetic element details. A detailed discussion of the storage will have to wait until later seminars. Discussion of the input and output will also be deferred. In particular, little has been said regarding the control. In this seminar I shall endeavor to discuss the control and to show something of how the computer will work.

First, let us approach the problem of determining needed elements by building up a system piece by piece, justifying each new element as it is added.

Consider Illustration A-30440. Every operation must be ordered. We need, therefore, a supply of orders within the machine. These orders can be kept in the element marked Storage. Suppose that this Storage is arranged as a number of storage places which we call registers. In order to facilitate obtaining the contents of these registers, let them be numbered consecutively from one up to the limit of storage capacity. Thus, each order will be put in some numbered register and can be obtained by having the storage read out the contents of the register having this number.

We also need some means of remembering these orders, that is, we must know what order to perform next. There are several methods available. It is possible to have each order contain information as to where in storage the next order is to be found. This method has certain advantages but is wasteful of storage space. Instead, consider the fact that orders are usually used in a definite sequence. This is not always so, in fact,

flexibility in the choice of what order is to be used is a very important ability of a computer, but the large majority of times orders are used one after the other. If, therefore, the orders are stored in the desired sequence in adjacent storage registers, they can be obtained by reading out of these registers in consecutive order.

Now, let us provide a counter which we will use to store the register number of the next order to be used. Normally, after each order is extracted and carried out, a one can be added to the counter. The next order to be extracted will then be the next in the storage sequence. It must be possible to change this sequence when desired. Changing the number in the counter will be considered later.

Illustration A-30441 shows how an order is set up. The counter holds the number of the register which holds the desired order. This register number is read out into the bus through a set of gate tubes CT. Attached to the storage must be a switch which can be used to pick the desired one of the storage registers. The order number travels down the bus and through another set of gate tubes to set this switch. The desired storage register has now been chosen.

Illustration A-30442 shows the next step, extracting the order from storage. The gate tubes on the storage are opened and the order read out into the bus. A problem then arises as to what to do with the order. As discussed in a previous seminar, the orders consist of two sections. One section is the operation order which could now be sent directly to the operation control. The other section is the storage register number which, since its purpose is to choose a number from storage, should be sent to the storage switch. It is possible to keep orders and numbers in two different storages but to do so would reduce the flexibility of the machine with few compensating advantages. It is preferable to have one storage which holds both orders and numbers and to allocate such sections of that storage as are desirable to each use. The storage switch has already been set to choose the register holding the order. The order cannot be sent directly to this switch since no time is available for clearing and preparing it for the reception of a new switch order.

A new register, the PR or Program Register, is provided for storing the order until the storage switch can be cleared. The same bus is used for the transfer. The gate tubes on the switch are not opened.

Illustration A-30443 shows how the operation is set up. The storage switch has been cleared. The order is read from PR into the bus and from the bus into the storage switch and operation control switch. The

operation control switch decodes the operation order and sets the operation control for the desired operation. The storage is set up to connect a particular register to the bus either for sending or receiving.

A complete arithmetic operation such as an addition requires three complete orders, two to send the numbers to be added to the arithmetic element, and one to store the result. It is possible to reduce this number if a result is to be used immediately, thus saving storing the result and then bringing it back to the arithmetic element for the next operation. In general, the first order will transmit one number to be operated on to the arithmetic element. Illustration A-30444 shows such an operation. The storage gate tubes are opened and the desired number read out into the bus. The input gate tubes of the arithmetic element are also opened allowing the number to go into the arithmetic element. Some additional operations such as clearing may be needed within the arithmetic element. The operation control will supply the necessary order pulses.

The next order will put the second number into the arithmetic element and order the operation to be performed. The entire sequence so far described must be gone through for each order. One is added to the counter. The counter contents are read into the storage switch via the bus. The order is read out into the program register and from there to the storage and operation control switches. The new number is read from storage to the arithmetic element where the desired operation is carried out under the control of the operation control.

Illustration A-30445 shows the storage of an arithmetic result. The output gate tubes of the arithmetic element are opened, the result transmitted to the bus and into storage via the storage input gate tubes.

All the orders described in an earlier seminar with four exceptions are described by the above sketches. The td order is the same as the storage order except that only part of the digits in the arithmetic element are transferred to storage.

The four exceptions are the shift right and left orders and the sub-program and conditional sub-program orders. The shift orders do not require transmitting a number from or to storage since the number to be shifted already lies in the arithmetic element. The usual procedure is followed, the storage switch is set, but the storage is not read.

The sub-program operations are shown in Illustration A-30446. The order has been extracted from storage and stored in PR. From PR the order has been transferred to the storage and operation control switches. As with the shift operations, the storage is set up but not read. The operation

control transfers the register position section of the order in PR to the counter. The next order to be taken will be the one residing in the register designated by the sub-program order.

The conditional sub-program order is shown in the same illustration. The operation control sends a pulse to the arithmetic element. If the sign of the number in the accumulator is negative as evidenced by a positive sign digit, (see the previous seminar on logic for a discussion of negative number representation), the pulse is stopped and nothing further occurs. The next order in sequence as given by the counter will be taken and the computation proceeds. If the number in the accumulator is positive, the pulse will be returned to the operation control switch where it will change the switch setting from conditional sub-program to sub-program. The register position section of the order in PR will be transferred to the counter as with the sub-program order and the order sequence will be changed.

So far today I have not discussed the mechanism by which the control orders the opening and shutting of gates and transfers numbers and information in the machine. This ordering can be accomplished by supplying pulses to the equipment at the right places and at the right times. The pulses to be supplied for setting up the order and operation are independent of the operation to be carried out. The pulses to be supplied in ordering the actual operation are dependent upon the operation control switch setting. We need, therefore a supply of pulses which appear on different lines at different specified times during the course of a computation period. These lines can then be connected to those parts of the equipment which are to receive pulses at the corresponding time. We need further, the ability to change these connections according to the operation to be carried out.

The basic device needed for a control of this type is an electronic switch. Such a switch is shown in Illustration A-30454. This switch, which is a simplified version of the types which will be used in the actual computer, consists of a number of flip-flop circuits whose plate loads are connected to B+ through a diode matrix. The switch shown contains only two flip-flops. As each of these flip-flops has two stable states, the switch has four possible positions. By adding more flip-flops, more positions may be obtained. The number of positions = 2^n where n = number of flip-flops.

The switch operates as follows. Consider the conditions under which only the right-hand tubes of a flip-flop are on. These tubes will be drawing plate current while the left-hand tubes of the flip-flops will be cut off and will be drawing no plate current. The tubes draw current through the diodes from B- via the load resistors at the left-hand side of the diagram. Examination of the diagram will show that plate current is being

drawn through only the three lower resistors. The top line of the switch is connected only to the left-hand tubes of the flip-flop. No current is drawn through the upper resistor and therefore the upper line will remain at B+. The lower three lines will be at a potential less than B+ by the drop in the load resistors. The upper line, therefore, is at a positive potential with respect to the lower three and can be used for controlling a gate tube. Suppose we change the condition of one of the flip-flops, say the left one so that the left-hand tube of the left-hand flip-flop draws current while the right-hand tube of the left-hand flip-flop does not. Plate current will be drawn through the upper resistor by the left-hand tube lowering the voltage of the upper line. It will be seen, however, that the third line is not connected to any of the on tubes. No current will be drawn through the corresponding resistor and line three will be at B+. Each of the four possible positions of the tube flip-flops results in a different output line being at B+. Diodes are used to prevent voltage feeding back to the on line and to keep all the off lines at the same potential.

Illustration A-30452 shows how a switch of this type can be used to make a time pulse distributor which will produce time pulses on different lines. This distributor consists of an eight-way diode matrix switch such as has just been described. The eight outputs are supplied to two-grid gate tubes while the flip-flops driving the switch are connected together in a counter circuit. Suppose now that all the flip-flops are in their off condition and that the corresponding condition of the switch is that the uppermost gate tube be on. A clock pulse comes in from the left, goes down through the connecting line and up to the grids of all the gate tubes. If the upper gate tube is on, a pulse will appear which is inverted by the transformer coupling and will go out as a positive pulse on the first line. If all the other gate tubes are off, no pulses will appear on the remaining seven lines. After the pulse has come out of the upper gate tube it will finish its travel through the delay line at the lower left and will go in and change the condition of the first flip-flop. This will change the switch position so that the upper gate tube will go off and the second gate tube will go on. The appearence of the next clock pulse therefore, will result in a pulse on the second line and no pulses on any other lines. The second pulse will change the first flip-flop back to a zero, the overflow from this flip-flop tripping the second flip-flop. In like manner each gate tube in turn is opened so that the corresponding clock pulses will come out in consecutive order starting with the top line and finishing with the bottom line. After the last pulse appears, the switch will be reset and the next pulse will appear at the top line. The distributor is receiving a continuous string of clock pulses and dividing them up so that they appear in consecutive order on these different lines. If, then, a complete operation is considered

to be started when the first clock pulse appears out of the time pulse distributor, it is only necessary to connect the different lines to those parts of the equipment which are to receive pulses at the corresponding time to have a simple control which will carry out the setting up of the order

Illustration A-30463 shows how a time pulse distributor can be used in conjunction with the operation control switch to order the different operations. The time pulse distributor is the same as that just described, the pulses appearing in consecutive order on lines one through eight The 32-position switch is the operation control switch which is supplied with the operation control part of the order Only one of the 32 possible outputs is selected at any given time

The row of gate tubes is used to connect the time pulses to the various parts of the equipment. These tubes are selected according to the position of the operation control switch.

Consider the leftmost gate tube. It is connected to the ca or clear and add order line of the operation switch and to this line only Therefore, whenever a ca operation is carried out this gate will be on and the No. 1 time pulse from the time pulse distributor will go out the line to whatever parts it is connected. In any other operation this gate tube will be off and no time pulse will go out The next pair of gate tubes have their outputs connected together and go to one particular part of the equipment. However, they supply time pulses at different times according to the operation to be carried out The left-hand gate tube is connected to the ca line while the right-hand gate tube is connected to the cs line. Therefore, when a ca operation is to be carried out, a time pulse will appear at time 2 on this line while when the cs operation is being carried out, a time pulse will appear at time 3. For all other operations, no time pulse at all will appear The fourth gate tube is connected to two of the operation lines, the add and subtract operations. Whenever an add or subtract order has been selected, a time pulse will appear at time 4 on the output of this gate, while on all the other operations no time pulse at all will appear. The last gate tube is like the first except connected to the ad order and time pulse No. 2. Actually for the machine there will be some forty or fifty of these gate tubes and the connection matrices will be quite large However, one 8-position time pulse distributor and a 32-position switch will be adequate for Whirlwind I control.

I would now like to describe the control necessary for carrying out a binary multiplication. Illustration A-30404 shows the modified binary multiplication described in one of the earlier seminars The sequence of

operations is as follows: examine the rightmost digit of the multiplier. Add in the multiplicand if this digit is a one. Following each examination, shift both multiplier and partial product to the right one digit. Again examine the new rightmost digit of the multiplier and add in the multiplicand to the shifted partial product if this digit is a one. Continue this process shifting multiplier and partial product at each step until every digit in the multiplier has been used. The product will now be complete.

The basic essentials for this multiplication are shown in Illustration A-30447. The multiplicand is stored in a binary register, the A-register, while the multiplier is stored in a second binary register, the B-register. The product is built up in the adding unit of the accumulator or AC. Gate tubes are provided for adding the contents of the A-register into the accumulator.

Illustration A-30448 shows the addition of a sensing device for examining the rightmost digit of a multiplier. A gate tube is connected to the rightmost digit flip-flop of the B-register in such a fashion that this gate tube is on when the digit of the rightmost column of the B-register is a one. A pulse is now supplied to this gate tube, and if the multiplier digit is a one, the gate tube between AR and AC will be opened and the contents of AR will be added to AC.

Illustration A-30449 shows the means provided for shifting the multiplier and partial product by one digit. A second gate tube is provided which is also connected to the rightmost digit of the B-register. This gate tube is arranged so that it is on when the rightmost digit is a zero. When a pulse is supplied to this gate tube the contents of the accumulator and B-register are both shifted to the right one digit. The digit which would ordinarily be shifted off the accumulator is put into the now empty space in the left-hand end of the B-register.

Illustration A-30450 shows a complete system for carrying out the successive additions of a multiplication. The clock pulses are supplied at the bottom line. If the rightmost digit of the multiplier is a zero, gate tube No. 2 will be on and gate tube No. 1 will be off. The clock pulse will pass through gate tube No. 2 and cause the partial product in AC and the multiplier in BR to be shifted to the right one digit. A new multiplier digit will be put in the sensing position.

If the rightmost digit of the multiplier is a one, gate tube No. 1 will be on and gate tube No. 2 will be off. The clock pulse will pass through gate tube No. 1 and add the contents of AR into AC. This pulse after passing gate tube No. 1 will also return to the input of the rightmost digit of BR changing it from a one to a zero. The next pulse to come along will find gate tube No. 2 on and will shift instead of add

At each step of the process, partial product and multiplier are shifted. The multiplicand is added to the partial product prior to the shift if the multiplier digit is a one. If the multiplier digit is zero, the shift only is performed.

Illustration A-30451 shows the complete multiplication control. A flip-flop is used to control a gate tube on the clock pulse input. When a multiplication is desired, a pulse is sent into this flip-flop turning on the gate tube and sending clock pulses to the gate tube control previously mentioned for forming the successive additions of the multiplication. A counter is supplied for stopping the additions when the multiplication is complete. At each shift this counter is indexed one point. When the counter is full, it will send an overflow pulse down to the input flip-flop turning off the clock pulse gate tube and stopping the multiplication. The counter is set so that this multiplication operation will be stopped after every digit of the multiplier has been sensed.

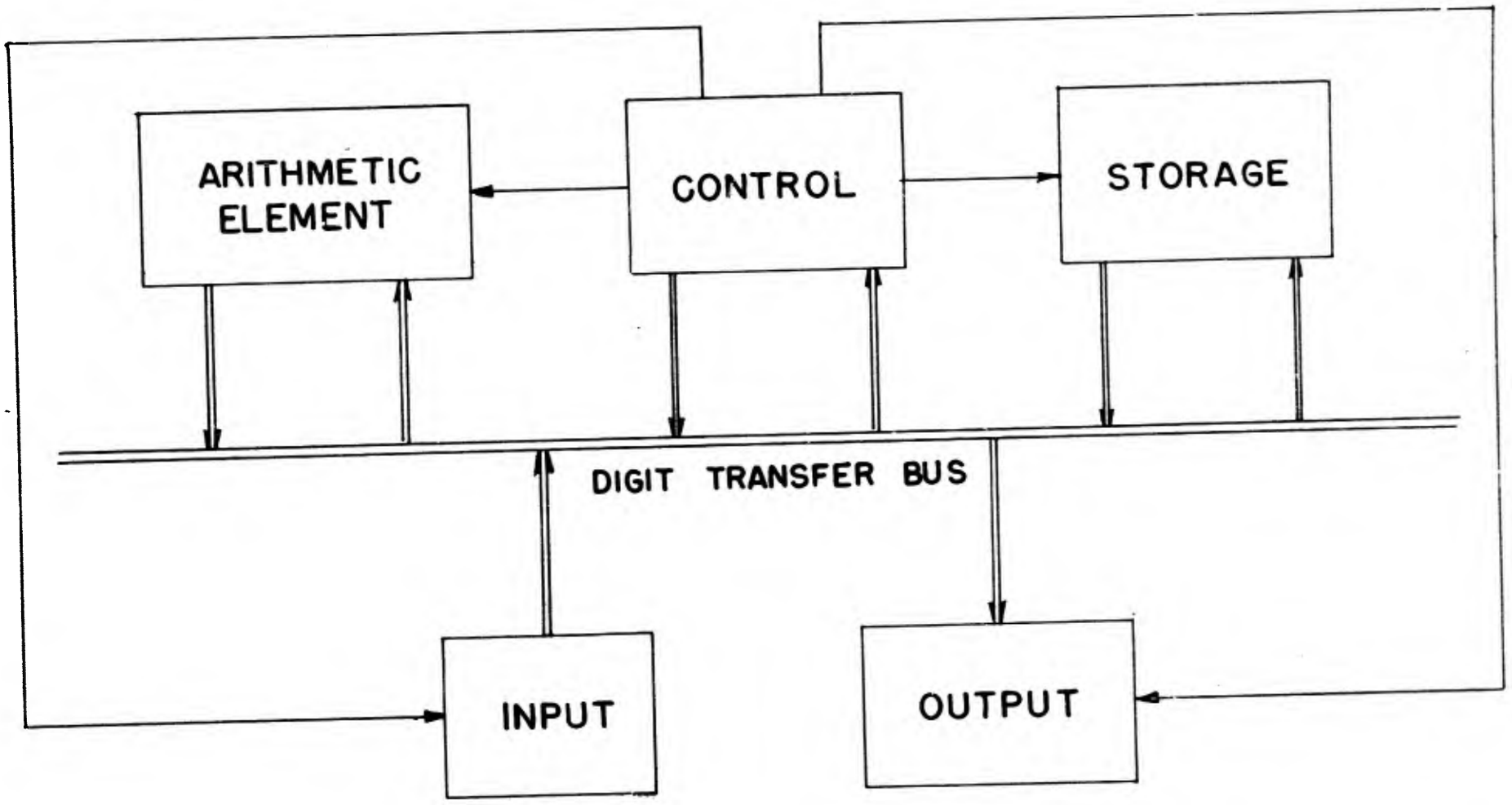
It would be possible to order the multiplication operation using the control distributor described previously. The main control is used for ordering some of the simpler operations such as addition and subtraction. More complicated operations such as multiplication and division may be more simply and efficiently accomplished by using all separate controls of the type mentioned.

Robert R. Everett

Robert R. Everett

RRE:has:vh

A-30339-1



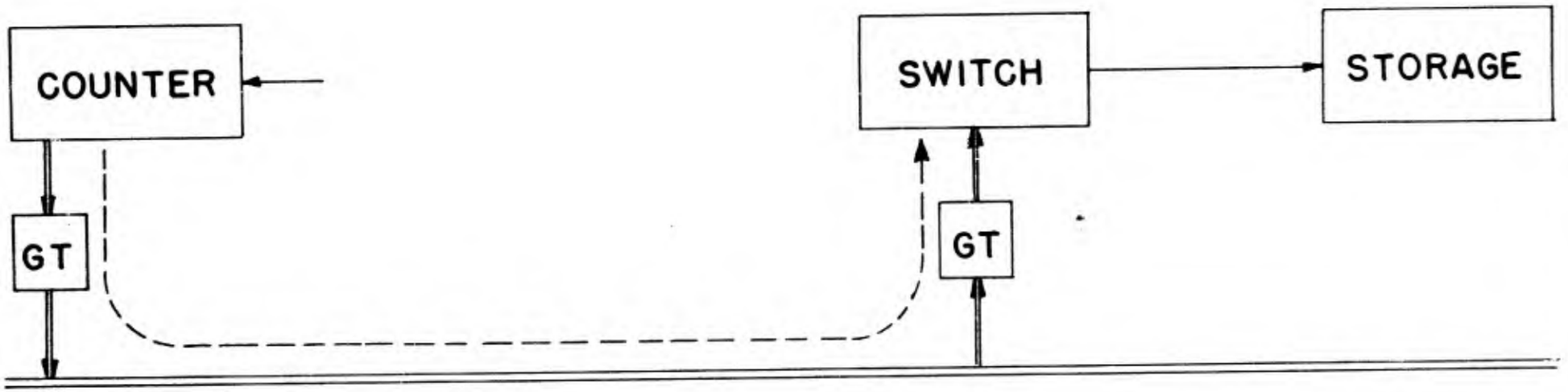
6345	RLK.
RICE	A-30339-1

A-30440

ORIGIN OF ORDERS

A-30440

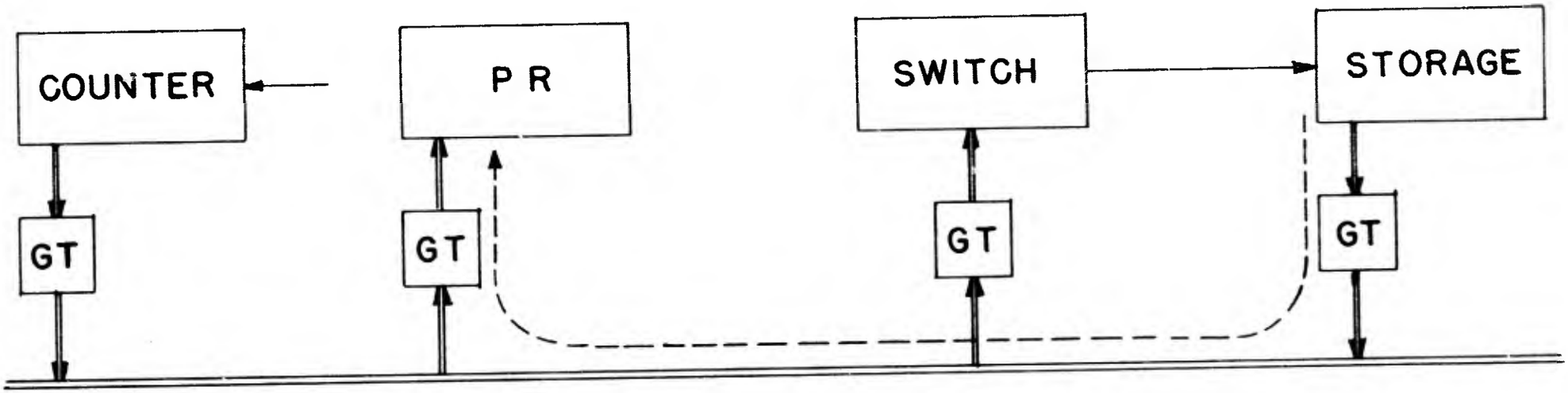
A-30441-1



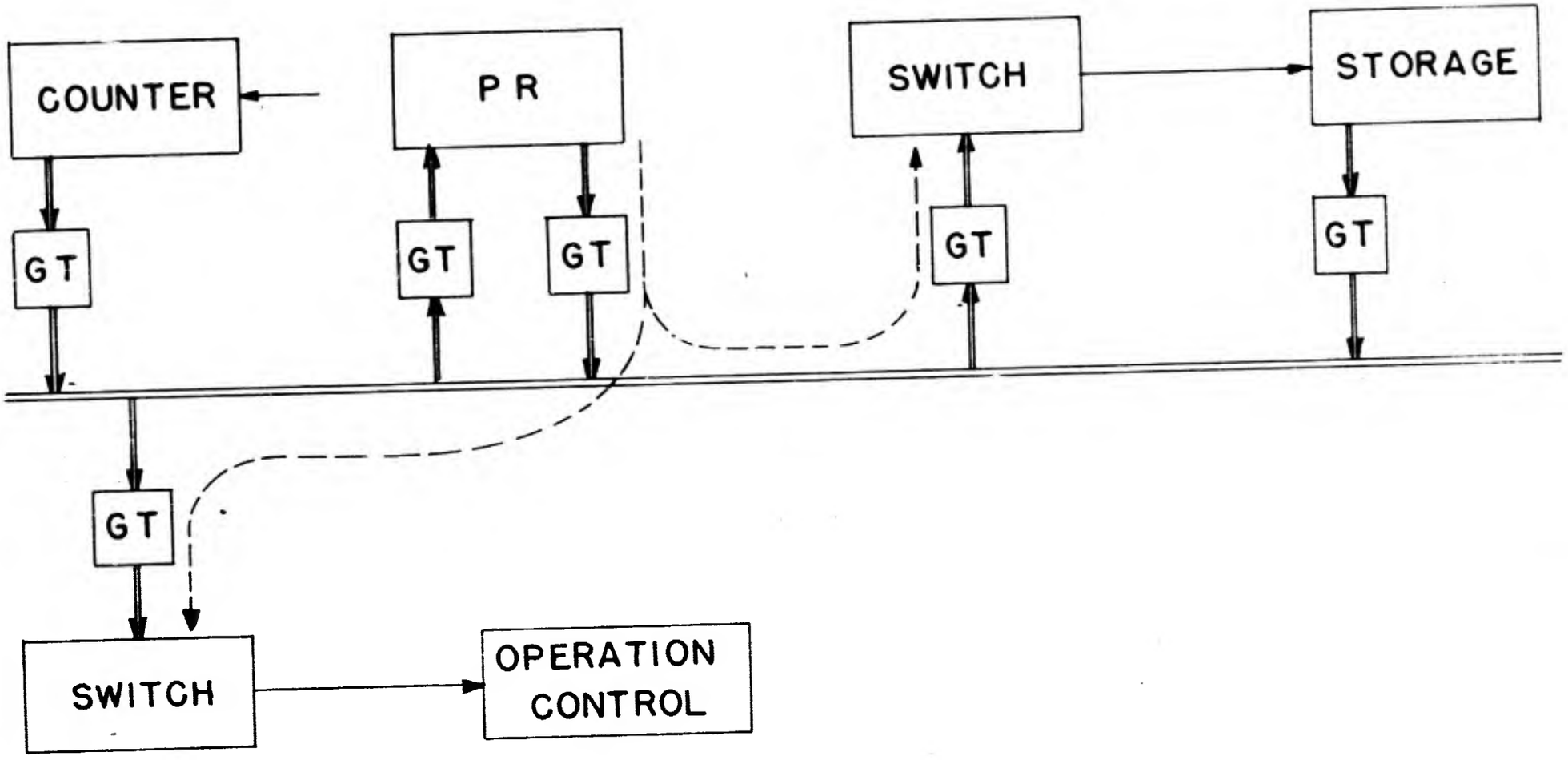
6345 R/K

A-30441-1

A-30442-1

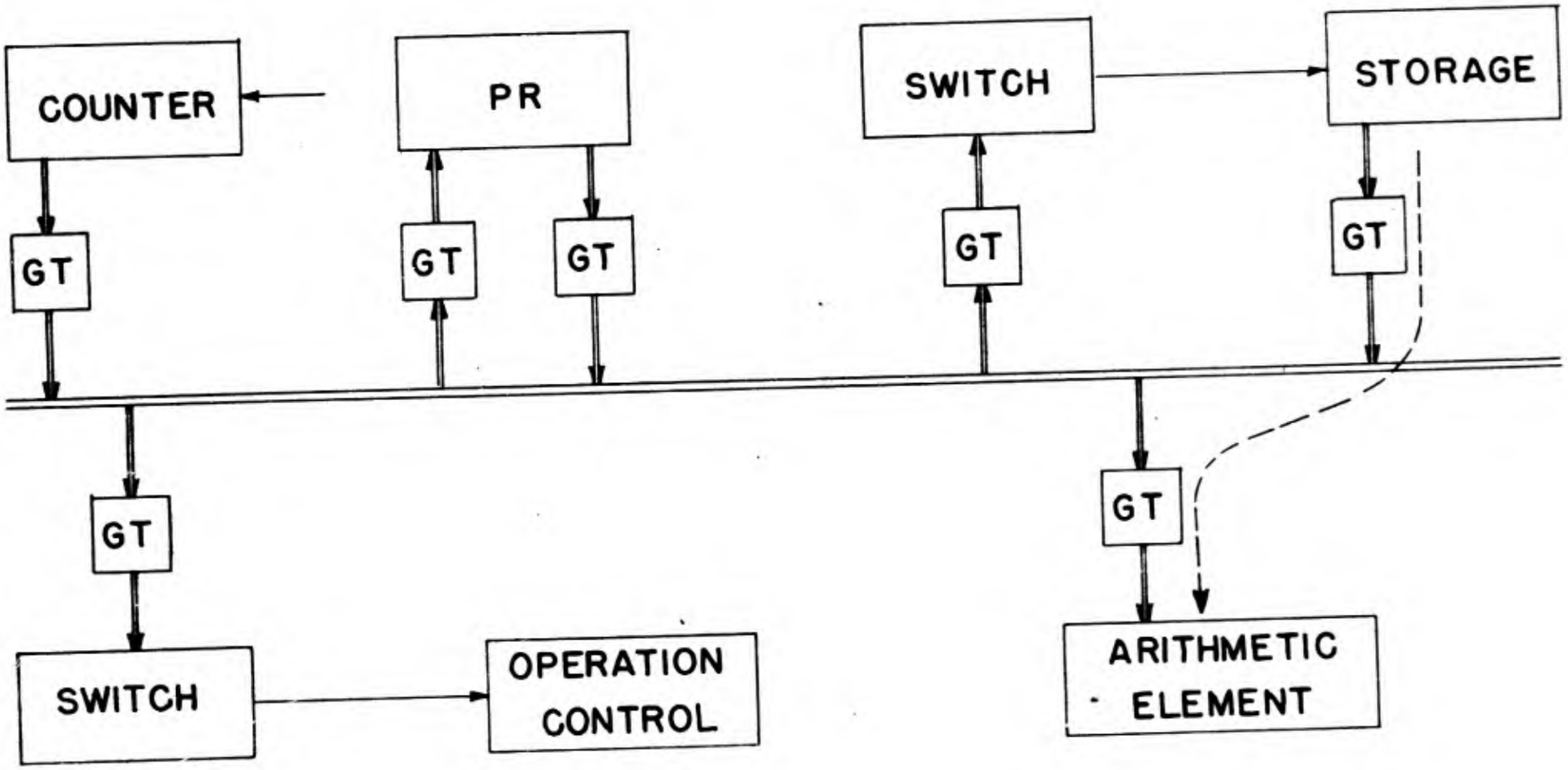


A-30443-1

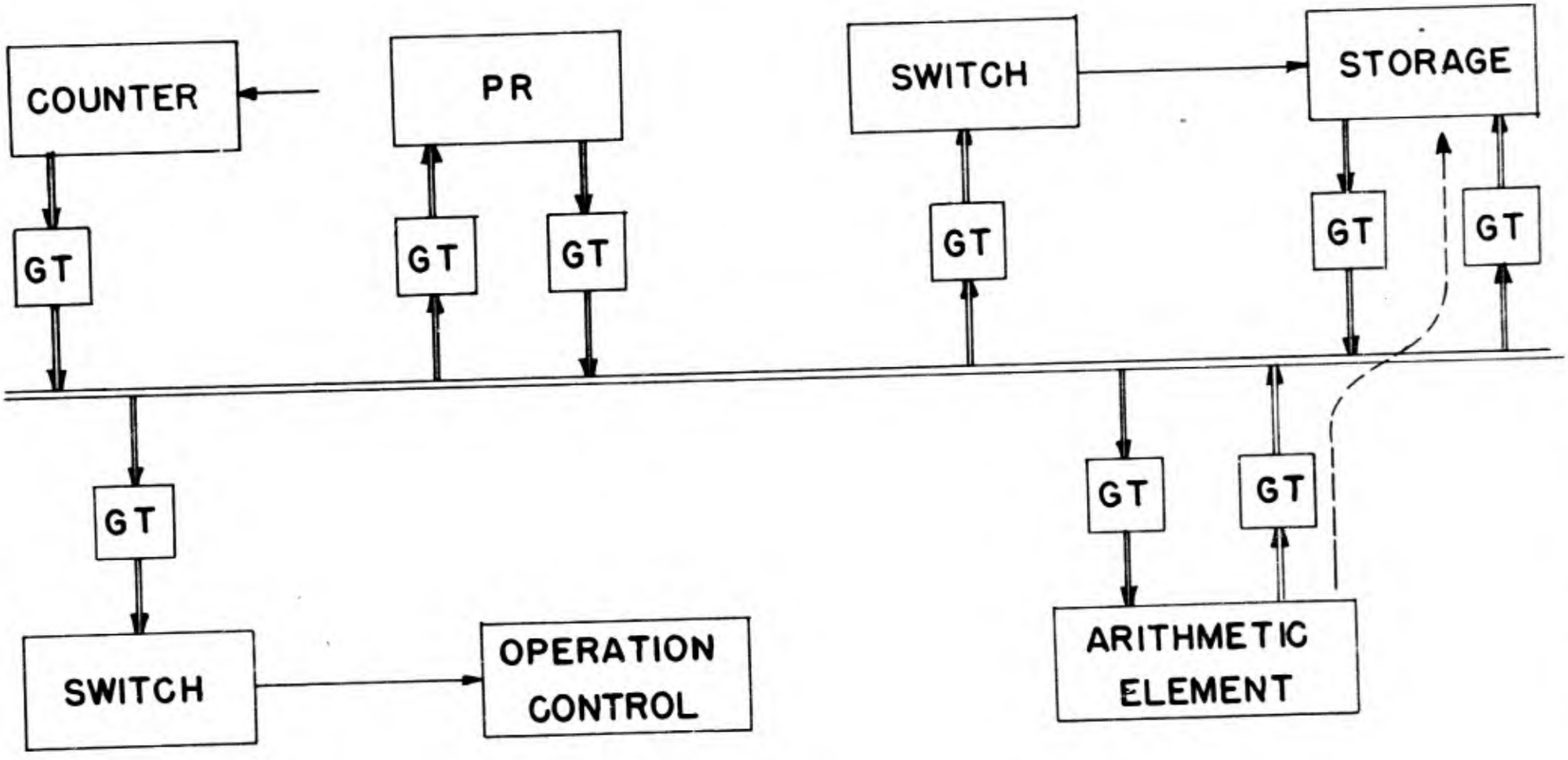


G345	B/K.
RRE	A-30443-1

A-30444-1



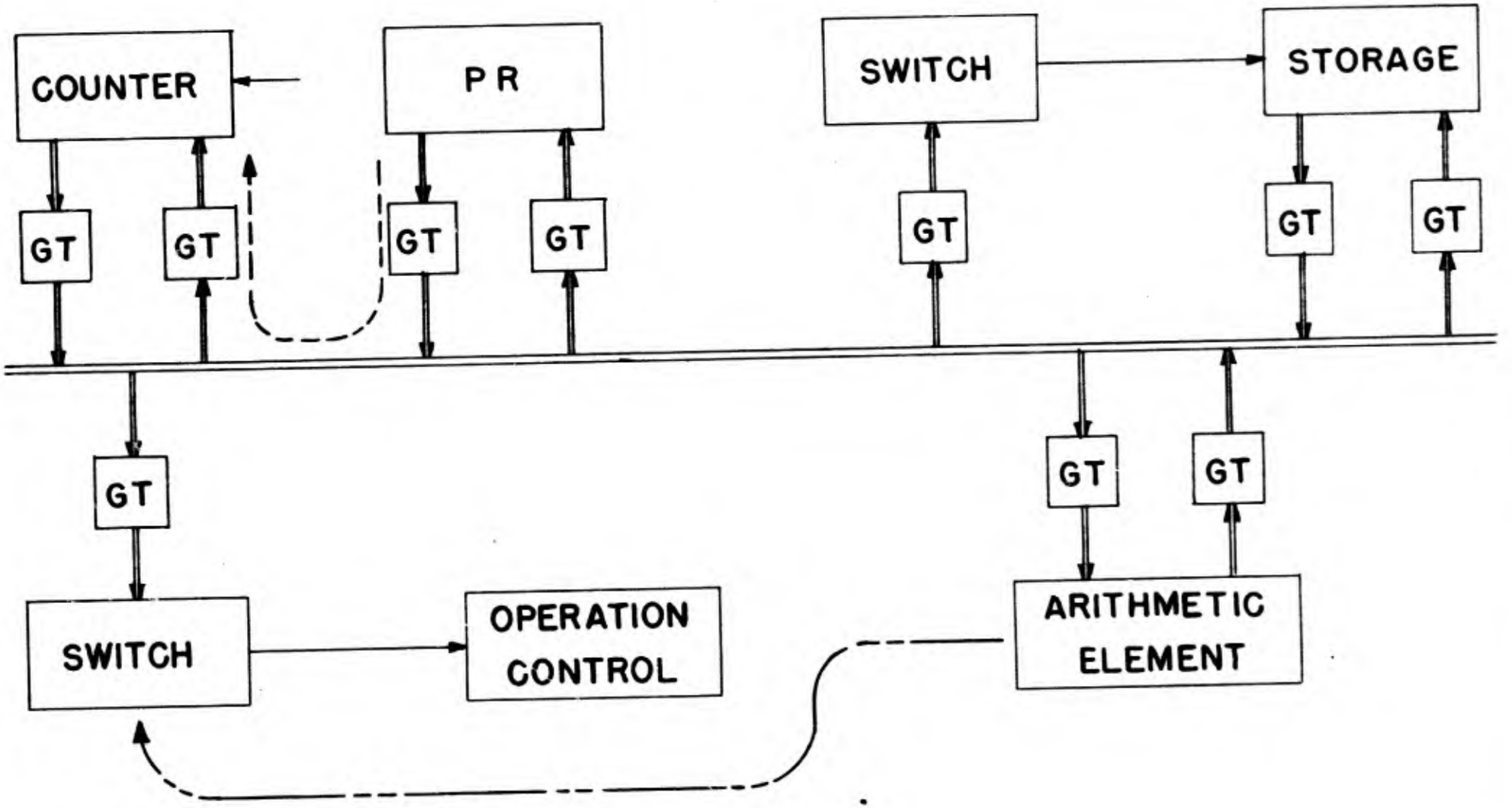
A-30445-1



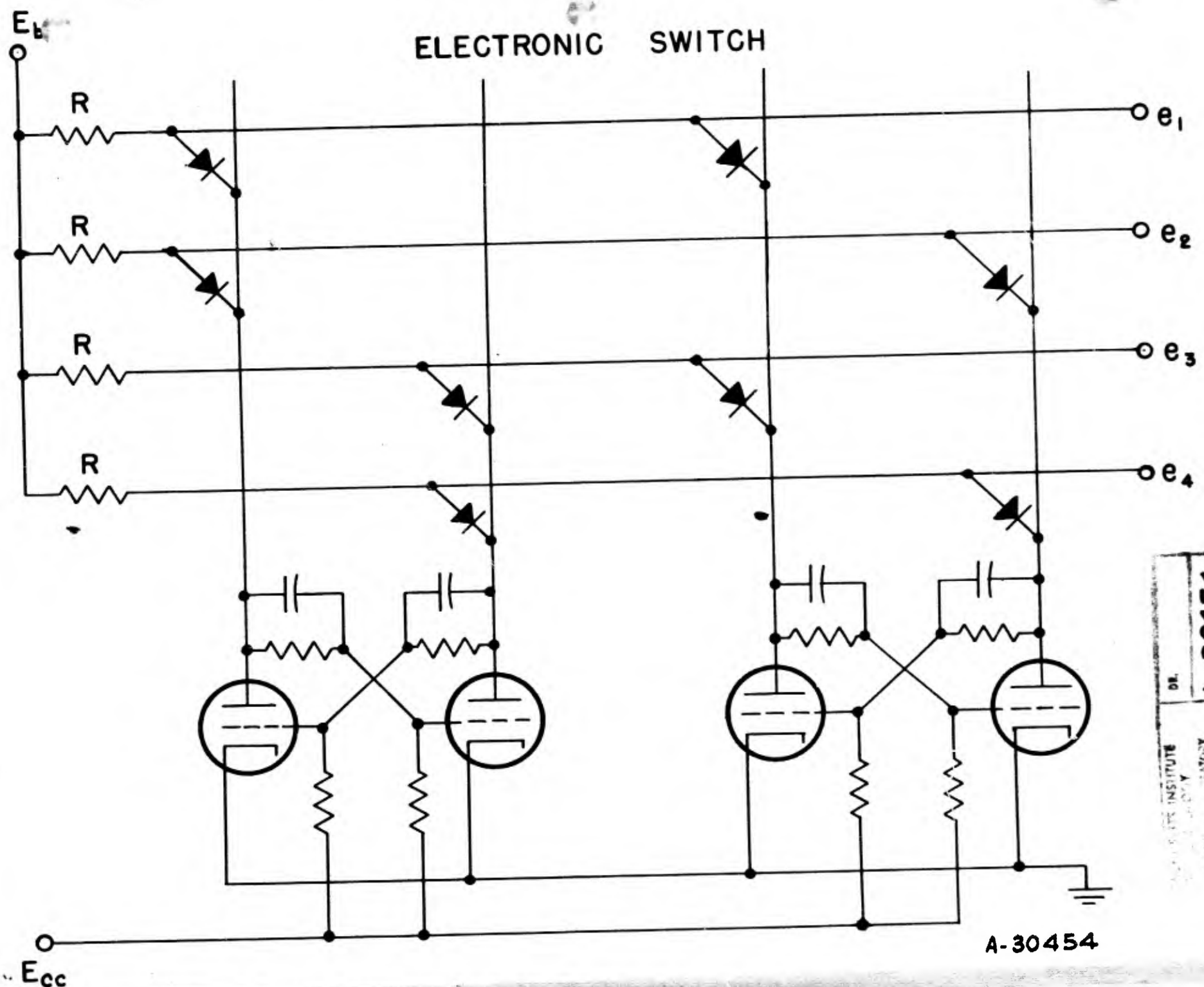
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

6345	R\K.
RICE	A-30445-1

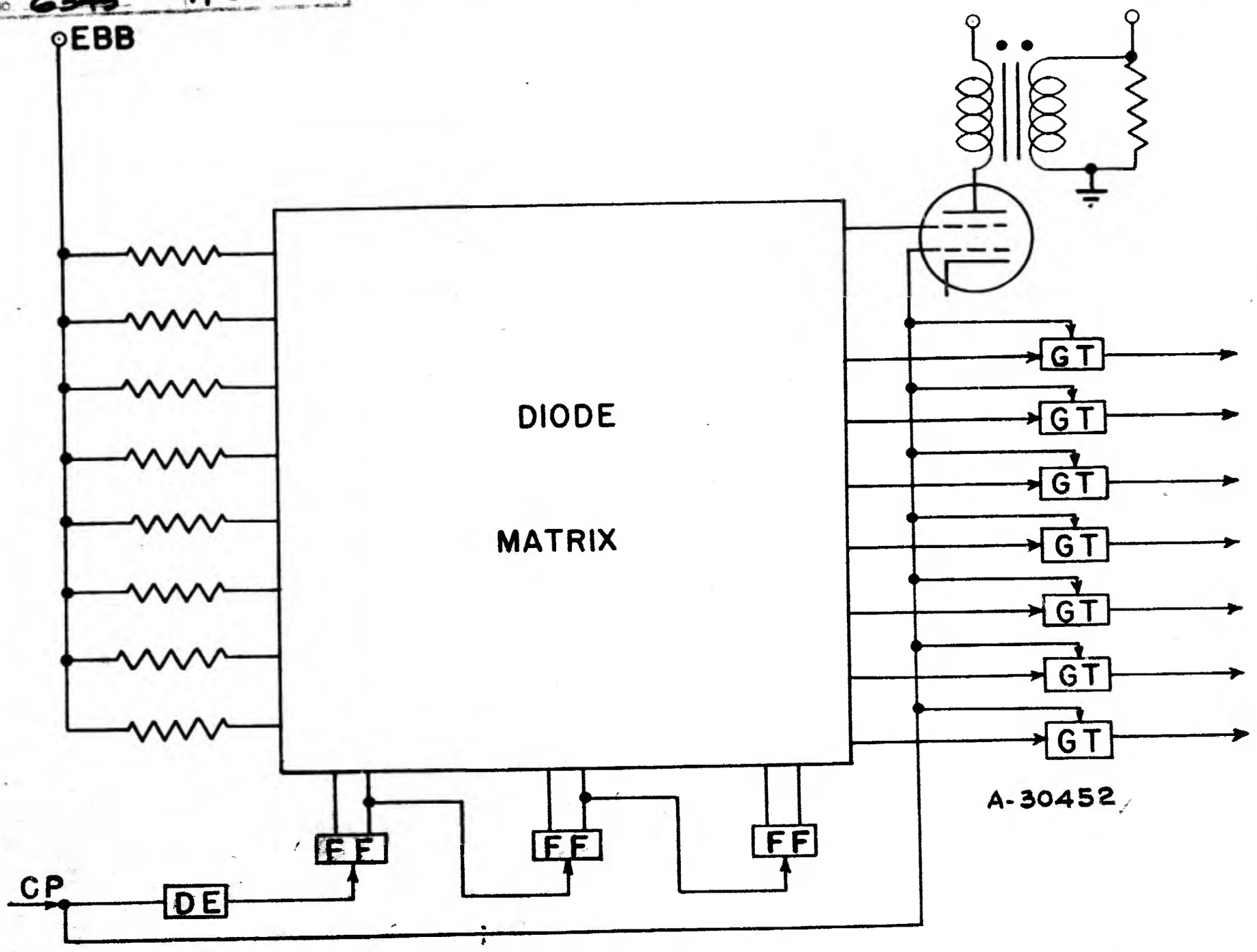
A-30446-1



6345	RK	
CRT		A-30446-1



TIME PULSE DISTRIBUTOR



A-30404-2

MODIFIED BINARY MULTIPLICATION

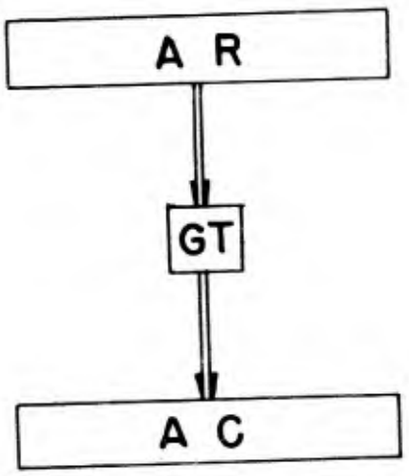
	10110	MULTIPLICAND
STEP 1	10011	MULTIPLIER
	10110	PARTIAL PRODUCT
	10110	MULTIPLICAND
	1001	SHIFTED MULTIPLIER
STEP 2	10110	SHIFTED PARTIAL PRODUCT
	10110	
	1000010	PARTIAL PRODUCT
	10110	MULTIPLICAND
	100	SHIFTED MULTIPLIER
STEP 3	1000010	SHIFTED PARTIAL PRODUCT
	00000	
	1000010	PARTIAL PRODUCT
	10110	MULTIPLICAND
	10	SHIFTED MULTIPLIER
STEP 4	1000010	SHIFTED PARTIAL PRODUCT
	00000	
	01000010	PARTIAL PRODUCT
	10110	MULTIPLICAND
	1	SHIFTED MULTIPLIER
STEP 5	01000010	SHIFTED PARTIAL PRODUCT
	10110	
	110100010	PRODUCT

6345 RRE

RLK 8/27/47

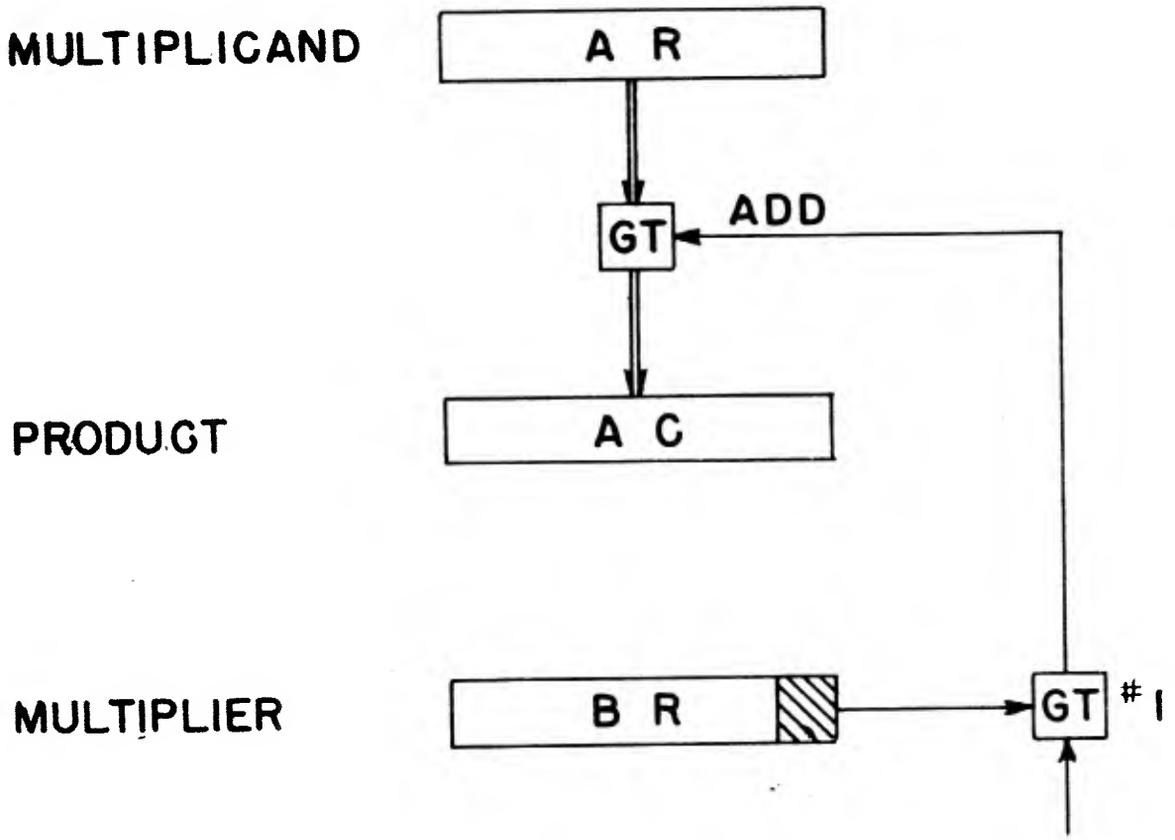
A-30404-2

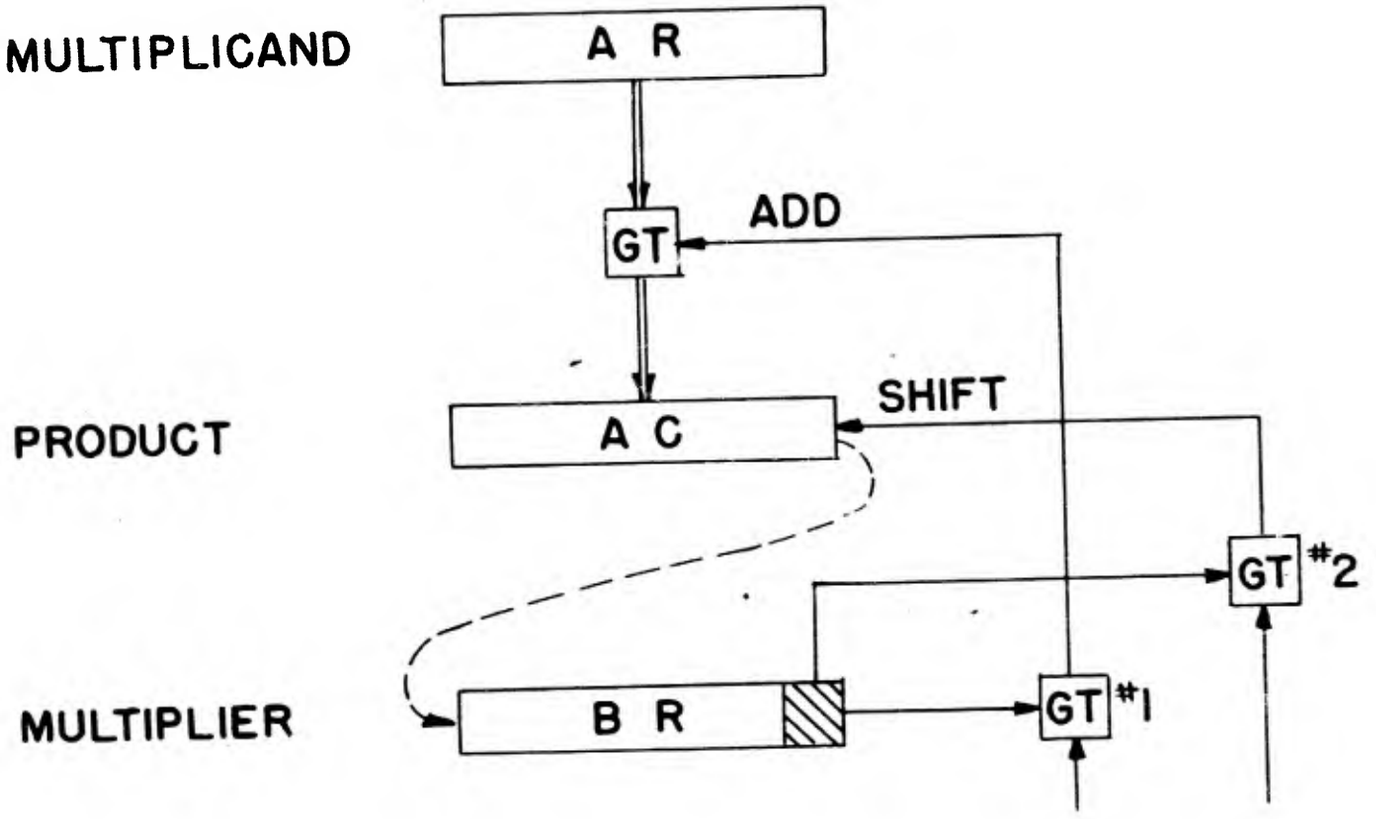
MULTIPLICAND



PRODUCT

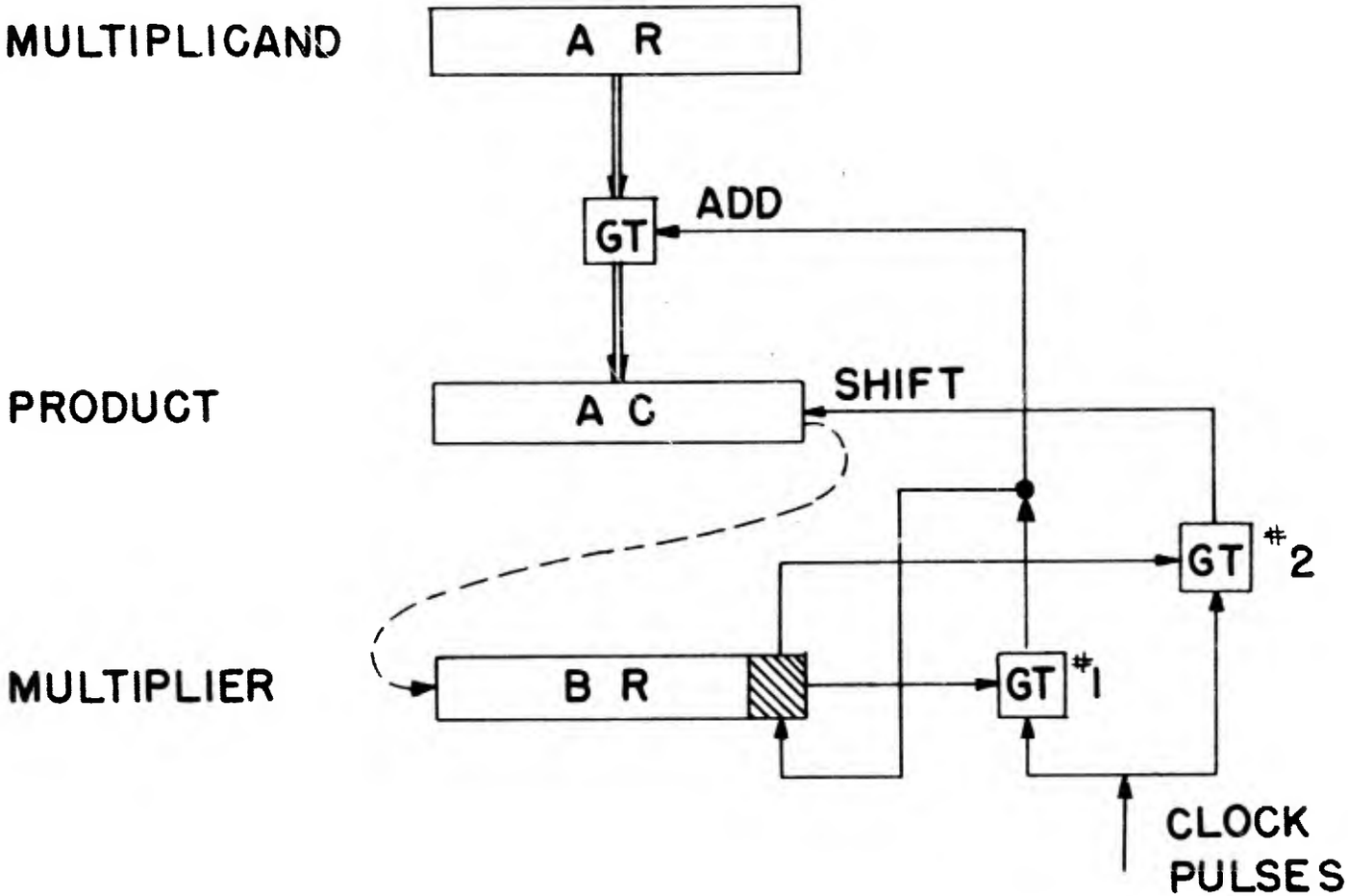
MULTIPLIER



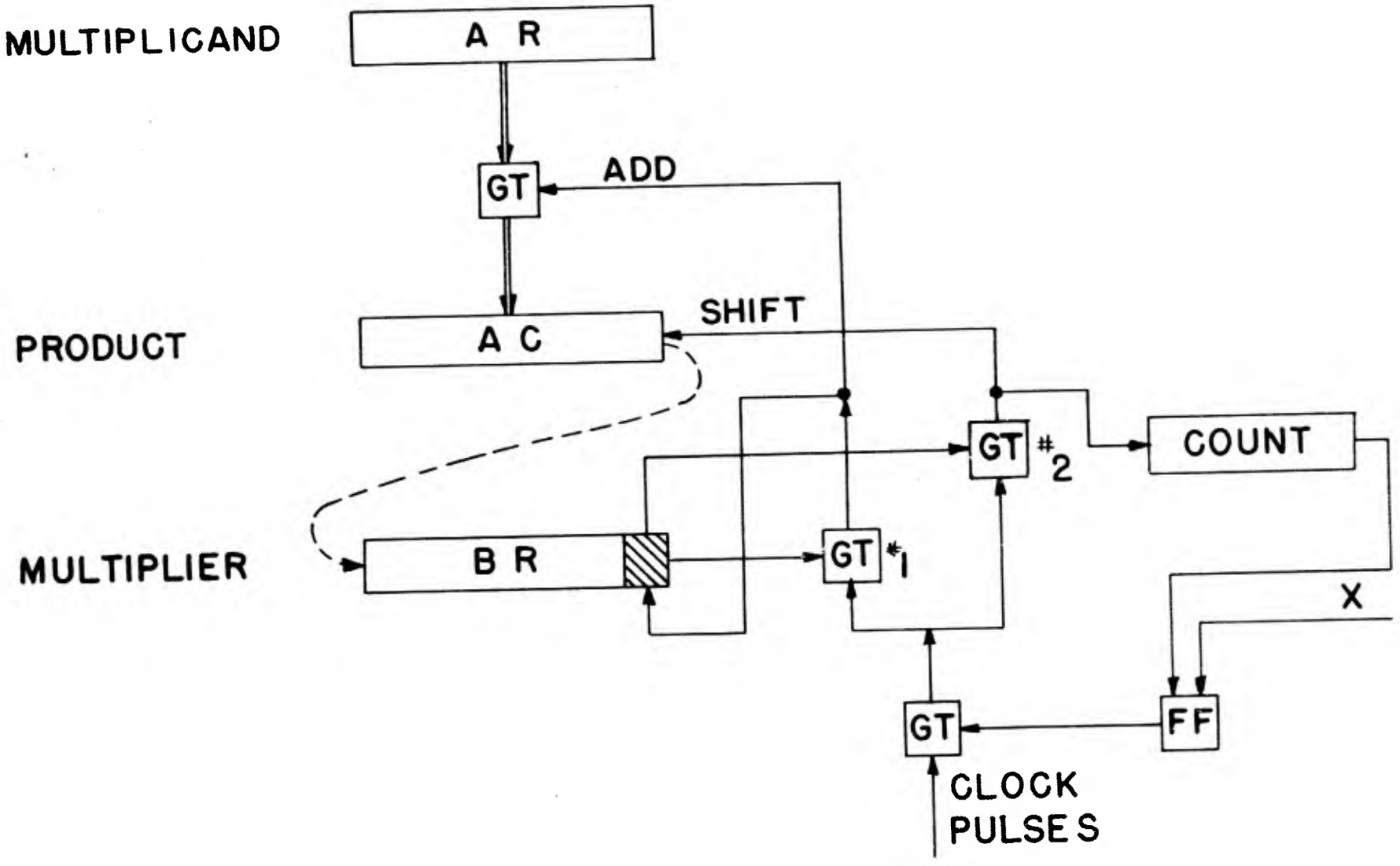

MULTIPLICAND
A R
ADD
GT
SHIFT
PRODUCT
A C
GT #2
MULTIPLIER
B R
GT #1

6345 RLK

A-30450-2



6345 R/K.
R/EE A-30450-2



6345 BLK.

A-30451-2

MEMORANDUM M-76

Servomechanisms Laboratory
Massachusetts Institute of Technology
Cambridge, Massachusetts

Written by: Jay W. Forrester 6345

Subject: Digital Computers in Science

Reference: "Machine Computation of Power Network Performance" by L. A. Dunstan, A.I.E.E. Technical Paper 47-87, December, 1946

Date: April 9, 1947

In the preceding seminar discussions we have considered the nature of electronic digital computers and have studied some of the circuits used for arithmetical computation. Block diagrams of a computer system have been outlined and the nature of some mathematical problems which will be encountered in numerical analysis have been indicated by Dr. Loud.

One should emphasize certain characteristics of electronic digital computers which have already been touched upon. Electronic computers of the type discussed will be practical only when used in such a manner that vast amounts of computation can be accomplished in proportion to the amount of manual setup and programming which is required. One way in which this can be accomplished was outlined by Mr. Everett in his discussion of generalized matrix multiplication. The generalized program for matrix multiplication involves only forty-seven control orders and is capable of multiplying together any two matrices of size within the capacity of the machine. The forty-seven orders are sufficient to control a computation requiring up to hundreds of thousands of operations. The matrix multiplication is an example of use of the machine for computation of its own controlling program. The outline of the problem to be accomplished has been given to the computer, but details in the carrying out of the individual steps are computed by the machine as it carries out the required numerical operations.

Engineering has now reached a viewpoint long held by the mathematician in indicating the solution of a problem. The mathematician often considers his problem solved when the form of the solution is known and when it can be written down in a compact symbolism. Matrix operations give us one example of this symbolic notation. Others can be found in the summation of series and in many forms of mathematical shorthand. Since this mathematical shorthand notation contains in it all of the information necessary for the solution of the problem, it is entirely possible to mechanize the

shorthand notation rather than the individual steps of the computation. The computer is, therefore, able to carry out a long series of operations the plan for which has been indicated only in outline or symbolic form. Generalized programs of the same type illustrated for matrix multiplication could equally well be arranged for the solution of ordinary linear differential equations or the evaluation of determinants and for the evaluation and summation of series operations.

In the field of engineering we can also expect that similar shorthand methods of setting up problems will come into use. Take, for example, the solution of alternating current network problems. Mr. L. A. Dunstan of The Federal Power Commission gave a paper at the last American Institute of Electrical Engineers' Convention in New York City describing the manner in which he had used punched card calculating machines for the solution of alternating current power systems. Mr. Dunstan stated that even with punched card equipment and the relatively large amount of manual operation involved, it was still possible to solve a-c power systems in approximately the same amount of time required on the network analyzer. He also stated that approximately one-half of the total time was required in the planning of the problem and in the selection of the circuit loops which were to be solved. It can be readily imagined, however, that with a sufficiently powerful and flexible digital computing machine a generalized program could be established for the solution of the a-c network problem. Such a solution might be set up on a nodal basis. After sufficient information was provided through the input mechanisms to completely describe the power system, it should be unnecessary for the operator to describe the exact method in which the solution will be accomplished.

A generalized program for solution of the a-c network system involves mathematical operation in complex quantities and is equally well suited to computations in the fields of vibration and stress analysis.

Every advantage possible must be taken of the large scale digital computing machine. It is not sufficient that one merely use it to replace the human operator of a desk type calculating machine. Since the electronic digital computer can follow almost any operation of control or computation which can be described, it is essential that one use the computer insofar as possible for the preparation of original data and for the correlation and interpretation of results of computation. As an example, consider the solution of ordinary differential equations with time as the independent variable. Where the solution of several dependent quantities is obtained as a function of time, it is often desirable that those quantities be plotted against one another rather than against the independent variable.

In such a problem the computer should be used for the required sorting, interpolating, and plotting of the result in the exact form desired for final use.

Many research programs in physics and mathematics are now blocked by the excessive numerical computing necessary to the next step of progress. Correlation studies in economics, medicine, and the social sciences must have the solution of large determinants which only high speed computers and a better understanding of the underlying mathematics can provide.

After a large scale computer has been in use for a period of time, libraries of control programs will be built up and become available for use on future work. Programs for the solution of many classes of problems can be prepared and other programs for special problems can often be built up as composite groupings of sections from programs on file.

A large scale electronic digital computer will be an expensive piece of equipment, and in order to be practical it should be kept in operation most of the time. Furthermore, continuous operation of such a computer will be highly desirable for the reduction of vacuum tube failures and for the resulting increased reliability and freedom from operating trouble. To provide sufficient computing load, it may be necessary to extend the sphere of usefulness of the large scale machine by long distance teletype connections. Methods are now being considered for teletype transcribing equipment for the remote transmission of problems to large scale computing equipment. Plans are being formulated for mathematical centers in several sections of the United States where large scale electronic computers can be located. To these will go problems from organizations with insufficient computing load to justify large scale machines and operating staff of their own.

Jay W. Forrester

JWF:vh

MEMORANDUM NO. M-54

6365

WRITTEN BY: Warren S. Loud

Page 1 of 7

SUBJECT: Mathematical Problems of High-Speed Digital Computation

Illustrations:
B-38171-G
A-30365
A-30362
A-30344
A-30366
A-30364
A-30455

DATE: March 26, 1947

Introduction

Today's discussion is concerned with mathematical problems connected with the use of high-speed digital computers and not with their design. As a consequence, there is little dependence of today's discussion on the talks given in the past two weeks by Mr. Forrester and Mr. Everett. This means in particular there will be no binary computation and no complicated program orders. All that is assumed is a machine which can do arithmetic or a program of arithmetic computation. The only properties of a Whirlwind computer which are different from a desk calculator are those of speed, and ability to follow a program automatically. It may seem premature to study problems of use of a machine when the machine is not yet in existence, but it appears that the machine will be able to handle problems involving more mathematics than we now know about.

I shall discuss mathematical problems from the point of view of errors arising in the numerical solution of a physical problem. When we solve a physical problem by numerical methods, we begin by formulating the problem in mathematical terms, usually differential equations or integral equations. We should notice here that this almost always involves an idealization of the problem. The true physical situation is too complicated to express in terms of mathematics which can be easily handled. We use differential equations because continuous mathematics is most easily handled with our present-day knowledge. However, it is possible to over-idealize a problem. It may well be that, though the physical problem has a solution, the idealized problem may fail to have a solution, or its solution may not tell the true physical situation. For example, we have worked with an oscillating system that exhibited sustained oscillations. If in the analysis of the system the equations were made linear, the mathematical solution failed to predict sustained oscillations, but would only predict damped oscillations. Thus, by over-simplifying the problem we fail to obtain even a qualitatively correct solution.

Once the problem is formulated mathematically, a numerical method of solution is devised and carried through. Again, approximations are made. A numerical process is a discrete process and as a result cannot possibly give the true solution of a continuous mathematical formulation. The result of such a process is a sequence of numbers which bears some relationship to the physical problem. Because of the many approximations made, this sequence of numbers is almost certain to be in error. There are three principal sources of error in the process described above. The first is improper mathematical formulation of the problem by over-idealization. The second source of error is the error which is deliberately introduced by using a discrete process to solve a continuous problem. Such errors are called truncation errors. They receive their name from the fact that when a numerical integration is performed, the value found is a truncation of the area represented by the integral. Most of the discussion of this talk will be concerned with truncation error. The third source of error is rounding off. At every intermediate step of the solution we have a number which is expressed with a finite number of decimal places. This is almost always an approximation, and thus introduces an error. Analysis of rounding-off errors is exceedingly difficult.

Discussion of Truncation Error

In the discussion of truncation error we will limit ourselves to very simple cases, cases where the exact solution is known. There are a number of reasons for restricting ourselves to such cases. First of all, the truncation error can be analyzed explicitly. Also, unless we know the true solution we cannot appraise the error well. The theory of differential equations provides theorems which will give an estimate of truncation error in very general cases. However, for a specific problem these theorems are much too crude. Still another reason for confining ourselves to problems where the exact solution is known is the temptation to accept a numerical solution as reasonably accurate when it is the only solution we have. The only way of obtaining a solution in many problems is a numerical method. However a numerical solution may be badly in error as we shall see. It is always the temptation to feel that the numerical solution is a good approximation to the true solution because it is the best we have. A set of numbers if printed neatly, looks very impressive, but a wrong set of figures looks just as impressive as a correct set.

Therefore, this afternoon we shall solve two differential equations: $\frac{dy}{dt} = y$ and $\frac{d^2y}{dt^2} = -y$. These are equations for which the exact solution is known. The solution of the first if y_0 is the value of y for $t = 0$, is $y = y_0 e^t$; the solution of the second, if $y = 0$ and $\frac{dy}{dt} = 1$ for $t = 0$ is $y = \sin t$. Let us consider the numerical solution of $\frac{dy}{dt} = y$. The steps are illustrated in illustration A-30455. We will find approximate values of y for $t = 0, h, 2h, 3h, \ldots, nh, \ldots$ To proceed from one step to the next, we assume that $\frac{dy}{dt}$ is constant in the interval considered and has the value which the differential equation prescribes at the beginning of the interval. In finding the value at $t = h$, we start with y_0 at $t = 0$, and assume that $\frac{dy}{dt}$ has the constant value y_0 for t between 0 and h. Under this assumption y_1, the value of y for $t = h$,

will be $y_0 + y_0 h$ or $y_0(1+h)$. To find the value of y for $t = 2h$ we assume that $\frac{dy}{dt}$ has the constant value y_1 for t between h and 2h. Thus the value y_2 at $t = 2h$ will be $y_1 + hy_1 = y_1(1 + h) = y_0(1 + h)^2$.

Continuing in this process it is seen that y_n, the value for $t = nh$, is $y_0(1 + h)^n$. By use of this formula we can see where the approximate solution is after any number of steps. Let us introduce the variable t by writing $n = \frac{t}{h}$. Using this we find that $y(t) = y_0\left[(1 + h)^{\frac{1}{h}}\right]^t$. We can see from this how the approximate solution tries to be $y_0 e^t$. We should remember from elementary calculus that the expression $(1 + h)^{\frac{1}{h}}$ approaches e as a limit as h approaches 0. Thus as our steps are taken shorter and shorter the approximate solution becomes more and more like the true solution. An important fact to notice about this approximate solution is that the ratio of the approximate solution to the true solution is equal to $\left[\frac{(1 + h)^{\frac{1}{h}}}{e}\right]^t$ which is an exponential function of time. This will be true it turns out, for any numerical solution of this type of a linear differential equation with constant coefficients.

The process just described is very crude. It can be improved to some extent by elaborating the process. For example, instead of making a straight line extrapolation of the solution as is done by assuming $\frac{dy}{dt}$ to be constant, we might make a parabolic extension of the solution by passing a parabola through the last two computed points of the solution with the proper slope at the last computed point. We might also improve the method by a numerical integration of the extrapolated solution. If the differential equation is written in the form $y_{n+1} = y_n + \int_{nh}^{(n+1)h} y\,dt$, where the straight line or parabolic extrapolation of y is used as the integrand, the value of y_{n+1} obtained is an improved approximation. An explanation of the reason for using such a simple process might be in order. It is true that more complicated numerical processes will give more accurate approximations. However, the philosophy adopted for numerical solutions has been that the machine is best suited to simple processes repeated rapidly many times rather than complicated processes repeated a lesser number of times. This utilizes the extremely high speed of the machine to best advantage.

Let us now consider the numerical solution of $\frac{d^2y}{dt^2} + y = 0$ of which the true solution is $y = \sin t$. In order to discuss the numerical solution of this equation, it is necessary to discuss damped oscillations and to introduce the concept of a damping ratio. A damped oscillation is the solution of a differential equation of the form

$$\frac{d^2y}{dt^2} + 2\zeta \frac{dy}{dt} + y = 0$$

ζ is called the damping ratio. If ζ is zero, the solution is undamped. If ζ is between zero and 1, the solution is underdamped. If ζ is equal to 1, the solution is critically damped, and if ζ is greater than 1, the solution is overdamped. Also if ζ is negative, the solution is negatively damped, which means the amplitude increases with time. If ζ is between zero and -1, the solution is an oscillation with exponentially increasing amplitude. For those of you who are more familiar with time constants than with damping ratios the damping ratio is equal to the reciprocal of the time constant in this case, where the undamped natural frequency is unity. Note that for negative damping the time constant is the time for the amplitude to grow to e times its original value.

To solve the equation $\frac{d^2y}{dt^2} + y = 0$ we form two simultaneous first order differential equations: $\frac{dy}{dt} = v$ and $\frac{dv}{dt} = -y$. The principal method used in solving these equations has been that of linear extrapolation and integration. The results of this process will be shown in some of the following illustrations. The method of solution is briefly the following. We write the two equations in the form

$$y_{n+1} = y_n + \int_{nh}^{(n+1)h} v\,dt$$

$$v_{n+1} = v_n - \int_{nh}^{(n+1)h} y\,dt$$

Thus to extend y the extrapolation of v is integrated numerically, while to extend v the extrapolation of y is integrated numerically. The results of this process for values of h equal to 1/8 and 1/10 of the period of the undamped vibration are shown in the first illustration, B-3817-G. Notice that the approximate solutions exhibit a negative damping characteristic. The solution having ten points per period is the better of the

two because of its shorter step length. The time constants of the two solutions are 17 seconds for the solution having 8 points per period and 33 seconds for the solution having 10 points per period. Since the total range of graph shown is about 32 seconds, it can be seen how these time constants actually are exhibited. In illustration A-30365 the numerical solutions are shown for some other step lengths. Notice that for the linear extrapolation and integration the apparent damping ratios are all negative. The method always brings an exponentially increasing truncation error. We might remark on the worst case, that of 2 points per period, which shows a damping ratio of -.515. This permits 16 time constant periods in the first 5 periods of the true solution. Thus the amplitude increases to e^{16} times its proper value in 5 periods. e^{16} is approximately ten million, so it can be seen just how bad truncation error can become.

I should like to consider next parabolic extrapolation and integration using 3 points and 1 slope. Of all the methods considered, this gave by far the best results. Illustration A-30362 shows a parabolic extrapolation. Illustration A-30344 shows the application of this parabolic extrapolation and integration to the solution of our differential equation. The shaded area in the figure represents the increments of y and v. Illustration A-30366 shows the results of using more complicated numerical methods to solve the equation $\frac{d^2y}{dt^2} + y = 0$. The first is where the linear extrapolation was determined by the two previous points of the solution rather than by one point and the prescribed slope. The second is the result of a combination of parabolic extrapolation by linear integration. The third is the best result obtained using parabolic extrapolation and integration. In each case there were twenty steps per period taken. Notice that the principal term of the approximation is the first in each case. The second term is always rapidly damped out. In each case the first term is very close to sin t, the frequency being approximately 1 and the coefficient in the exponent being very small.

We also attempted the numerical solution in the case of an equation with a positive damping ratio. In the undamped case we found that if the number of points per period was 2 or 3, the numerical solution always exhibited a negative damping characteristic. We found also that even though the equation had a positive damping characteristic, the numerical solutions always exhibited a negative damping characteristic if the number of steps became less than 3 per period. This is shown in illustration A-30364 which shows apparent damping ratio as a function of points per period for various positive damping ratios in the equation solved. Notice that for more than 10 points per period the curves do approximate their proper values quite closely but that for less than

5 points per period, all of the curves become negative showing that the negative damping characteristic will always be present. This illustration has been drawn for the case of linear extrapolation and integration. However, the same characteristic will be present with any method of extrapolation and integration among those which have been described.

Another idea which was tried was the numerical solution of an equation which had two natural frequencies, one 10 times the other. A step length was chosen equal to 1/20 of the longer period and equal to 1/2 of the shorter period. Then the numerical method was tried. Initial conditions were chosen so that the high frequency term would be completely suppressed in an exact solution. Unfortunately the numerical solution did not suppress the high frequency term and it appeared with all its wild diversions in the numerical solution, obliterating the low frequency term quite completely.

In conclusion, I should like to make a few remarks about the other two sources of error. First, consider rounding-off error. Rounding-off error is of importance with high-speed calculating machines. In a normal computation the number of rounding-offs is small, but in a numerical solution using a high-speed machine there will be many thousands of rounding-offs which will introduce considerable rounding-off error. Professor Rademacher of the University of Pennsylvania has studied rounding-off error. His result is that rounding-off error increases like $\frac{1}{\sqrt{h}}$ as h grows small.

With regard to errors due to improper formulation, we see the necessity for existence proofs. The fact that a physical problem has a solution does not justify the conclusion that the idealized problem has a solution. A differential equation may not represent the true physical picture all the time. For example: suppose in formulating a problem we wrote the differential equation $\frac{dy}{dt} = 1 + y^2$ and we desired the solution of this equation from t = 0 to t = 5. We would find that the numerical method of solution would diverge. Experience with linear equations would suggest that the step length of the process should be shortened. However, with this equation no matter how much the step length is shortened, the divergence will persist. The reason for the divergence is not within the numerical method. The reason lies in the differential equation itself. The solution of this equation is y = tan t, and this solution is bound to become infinite at least once between zero and 5.

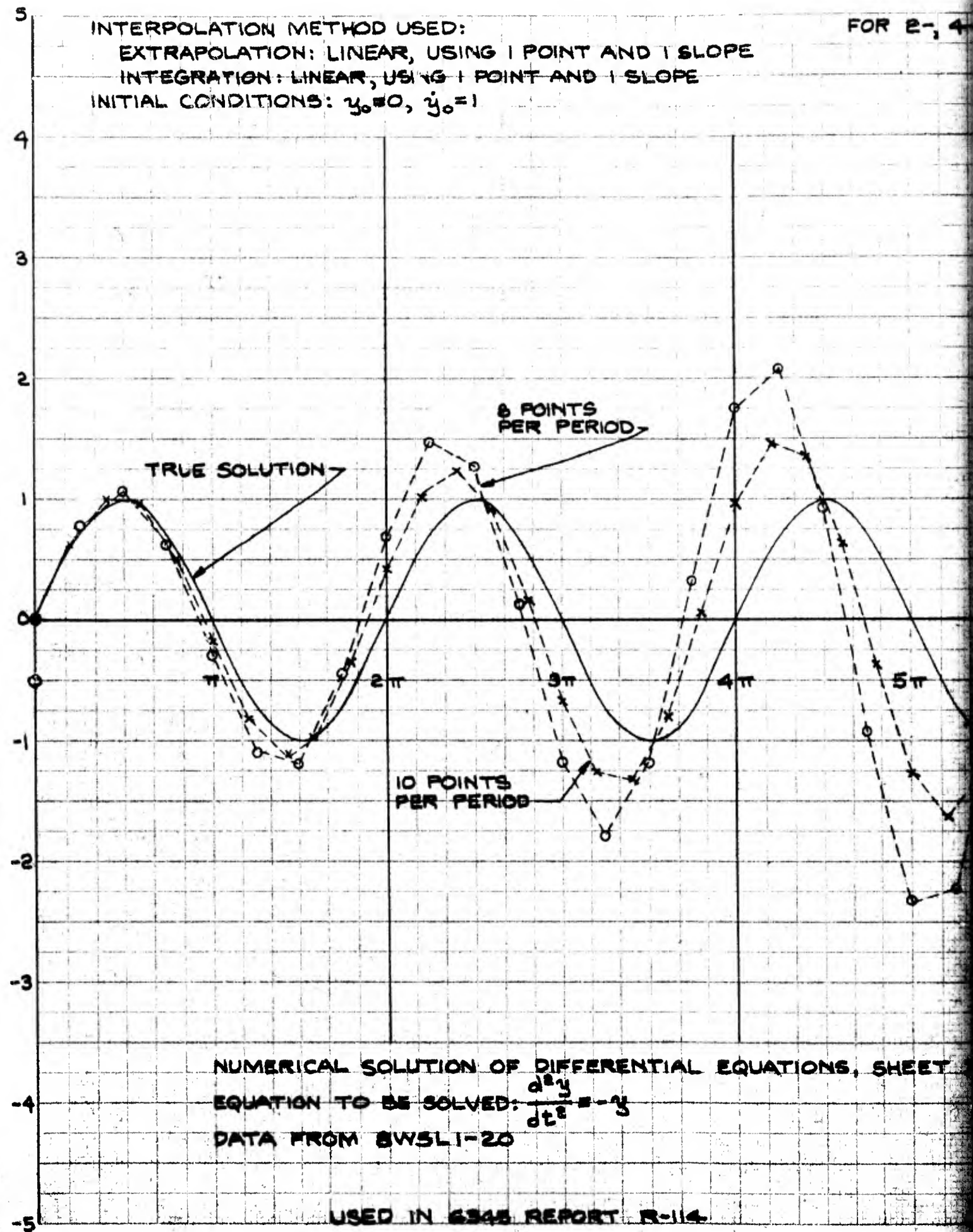
Warren S Loud

Warren S. Loud

WSL:vh

INTERPOLATION METHOD USED:
EXTRAPOLATION: LINEAR, USING 1 POINT AND 1 SLOPE
INTEGRATION: LINEAR, USING 1 POINT AND 1 SLOPE
INITIAL CONDITIONS: $y_0=0$, $\dot{y}_0=1$

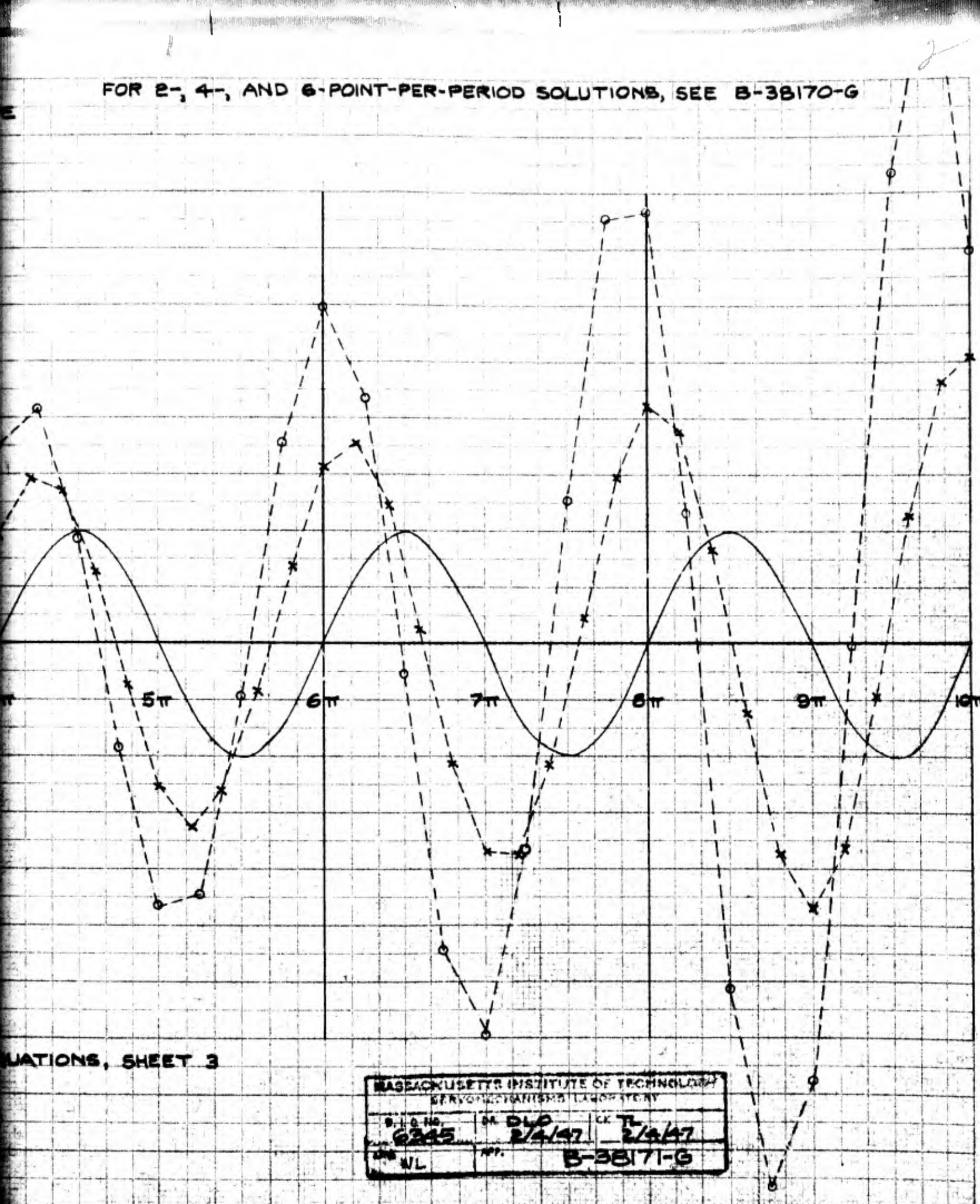
FOR 2-, 4



NUMERICAL SOLUTION OF DIFFERENTIAL EQUATIONS, SHEET
EQUATION TO BE SOLVED: $\frac{d^2y}{dt^2}=-y$
DATA FROM BWSL 1-20

USED IN 6345 REPORT R-114.

FOR 2-, 4-, AND 6-POINT-PER-PERIOD SOLUTIONS, SEE B-38170-G



UATIONS, SHEET 3

MASSACHUSETTS INSTITUTE OF TECHNOLOGY SERVOMECHANISMS LABORATORY		
S. L. O. NO. 6245	DR. DLP 2/4/47	CK. TL 2/4/47
JOB WL	APP. B-38171-G	

KEUFFEL & ESSER CO. N. Y.

$$\frac{d^2y}{dt^2}+y=0\,;$$ TRUE SOLUTION: $y = \sin t$

NUMERICAL SOLUTION

LINEAR ONE-POINT ONE-SLOPE EXTRAPOLATION AND INTEGRATION

NO. OF POINTS PER PERIOD	NUMERICAL SOLUTION	APPARENT DAMPING RATIO
20	$y = e^{.00382t} \sin 1.01592t$	−.00382
10	$y = e^{.0304t} \sin 1.05711t$	−.0304
8	$y = e^{.0578t} \sin 1.08079t$	−.0578
6	$y = e^{.1256t} \sin 1.11113t$	−.126
4	$y = e^{.2945t} \sin 1.09401t$	−.294
2	$y = e^{.5145t} \sin .78553t$	−.515

A-30365

2 - POINT: ONE SLOPE EXTRAPOLATION

$$Y = ax^2 + bx + c$$

$$Y_n = ax_n^2 + bx_n + c$$

$$Y_{n-1} = a(x_n - h)^2 + b(x_n - h) + c$$

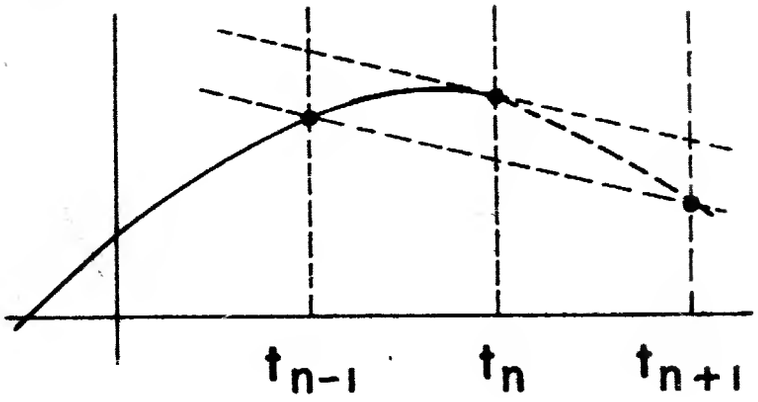
$$\dot{Y}_n = 2ax_n + b$$

$$X_n = 0$$

$$Y_n = c$$

$$Y_{n-1} = ah^2 - bh + c$$

$$\dot{Y}_n = b$$



$$a = \frac{Y_{n-1} + bh - c}{h^2} = \frac{Y_{n-1} + \dot{Y}_n h - Y_n}{h^2}$$

$$Y_{n+1} = \left(\frac{Y_{n-1}}{h^2} + \frac{\dot{Y}_n}{h} - \frac{Y_n}{h^2} \right) h^2 + \dot{Y}_n h + Y_n$$

$$= Y_{n-1} + h\dot{Y}_n - Y_n + \dot{Y}_n h + Y_n$$

$$= Y_{n-1} + 2h\dot{Y}_n$$

A-30362

NUMERICAL SOLUTION

$$\ddot{y} = -y$$

KNOWN: $\ddot{y}, \dot{y}, y$, at t_n and t_{n-1}

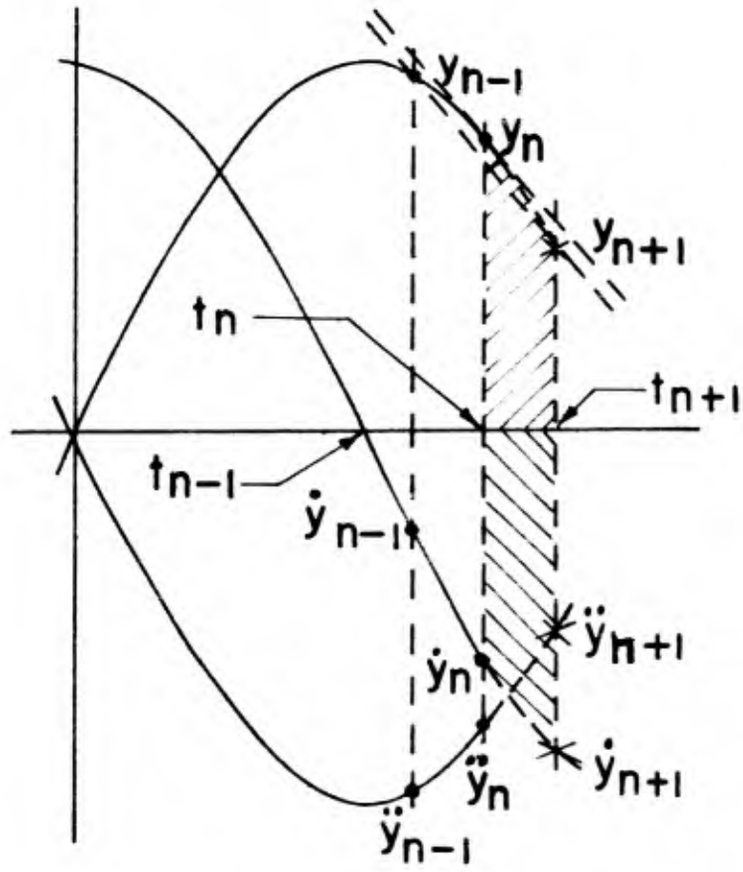
$$y_{n+1} = y_{n-1} + 2h\dot{y}_n$$

$$\Delta\dot{y} = -\frac{5y_{n+1} + 8y_n - y_{n-1}}{12}h$$

$$\Delta\ddot{y} = -\frac{5\dot{y}_{n+1} + 8\dot{y}_n - \dot{y}_{n-1}}{12}h$$

$$y_{n+1} = -\ddot{y}_{n+1}$$

A-30344



$$\frac{d^2y}{dt^2} + y = 0 \qquad \text{TRUE SOLUTION: } y = \sin t$$

NUMERICAL SOLUTION FOR TWENTY POINTS PER PERIOD
WITH APPARENT DAMPING RATIO

(1) LINEAR TWO-POINT EXTRAPOLATION AND INTEGRATION

$$y = e^{+.0095t}\left[.0121\cos(1.04t) + 1.0459\sin(1.04t)\right] \qquad \zeta = -.00913$$
$$+ e^{-5.9t}\left[-.0121\cos(3.956t) - .0459\sin(3.956t)\right]$$

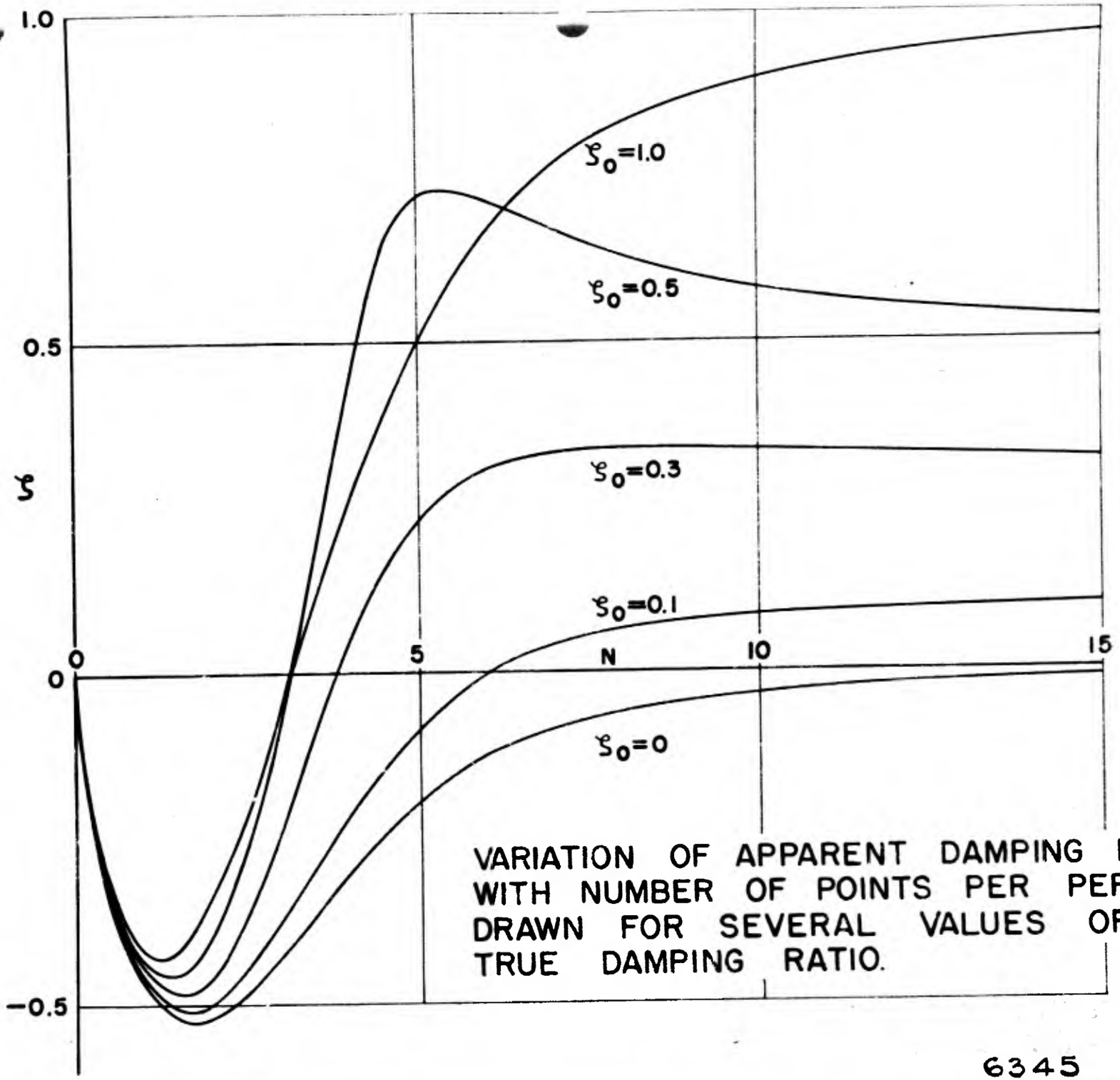
(2) PARABOLIC TWO-POINT ONE-SLOPE EXTRAPOLATION; LINEAR ONE-POINT ONE-SLOPE INTEGRATION

$$y = e^{-.0053t}\left[-.0129\cos(.995t) + 1.0446\sin(.995t)\right] \qquad \zeta = +.00533$$
$$+ e^{-5.886t}\left[+.0129\cos(5.995t) + .0446\sin(5.995t)\right]$$

(3) PARABOLIC TWO-POINT ONE-SLOPE EXTRAPOLATION AND INTEGRATION

$$y = e^{-.00256t}\left[-.0136\cos(1.0012t) + 1.045\sin(1.0012t)\right] \qquad \zeta = +.00256$$
$$+ e^{-7.211t}\left[+.0136\cos(6t) + .045\sin(6t)\right]$$

A-30366

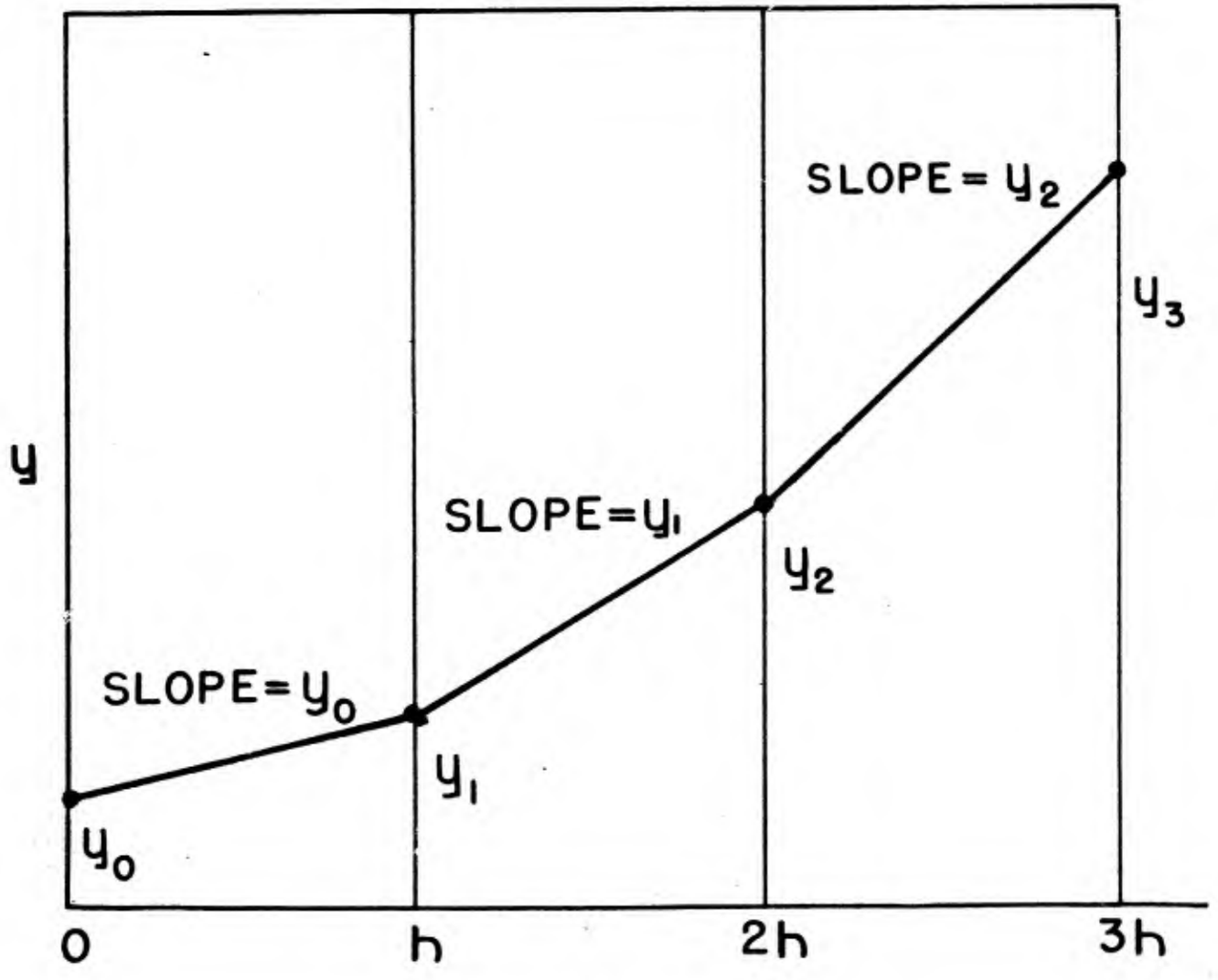


VARIATION OF APPARENT DAMPING RATIO WITH NUMBER OF POINTS PER PERIOD, DRAWN FOR SEVERAL VALUES OF TRUE DAMPING RATIO.

A-30364

A-30455

NUMERICAL SOLUTION OF $\frac{dy}{dt} = y$



$$\frac{dy}{dt} = y$$

$$y_{n+1} = y_n(1+h)$$

$$y_n = y_0(1+h)_0^n$$

6345 WL

TH 4/7/47 — TL 4/7/47

A-30455

6345
Report 80

SERVOMECHANISMS LABORATORY
Massachusetts Institute of Technology
Cambridge, Massachusetts

Date of Report: January 15, 1946 Page 1 of 5 Pages

Subject: The Binary System of Numbers

Identification: The representation of decimal numbers in the binary system and the processes of binary arithmetic are explained.

Discussion:

Representation of Numbers - The decimal system takes its name from the fact that it is based on ten digits (0, 1, ... 9,) and all numbers are composed of those 10 digits. The binary system, analogously, takes its name from the fact that it is based on 2 digits, (0, 1.) and all numbers in the binary system are made up of those 2 digits. The decimal system has a base of 10, the binary system has a base of 2.

Decimal System	Equivalence	Binary System	Equivalence	
1	$1x10^0$	1	$1x2^0$	= 1
2	$2x10^0$	10	$1x2^1 + 0x2^0$	= 2
3	$3x10^0$	11	$1x2^1 + 1x2^0$	= 3
4	$4x10^0$	100	$1x2^2 + 0x2^1 + 0x2^0$	= 4
5	$5x10^0$	101	$1x2^2 + 0x2^1 + 1x2^0$	= 5
6	$6x10^0$	110	$1x2^2 + 1x2^1 + 0x2^0$	= 6
7	$7x10^0$	111	$1x2^2 + 1x2^1 + 1x2^0$	= 7
8	$8x10^0$	1000	$1x2^3 + 0x2^2 + 0x2^1 + 0x2^0$	= 8
9	$9x10^0$	1001	$1x2^3 + 0x2^2 + 0x2^1 + 1x2^0$	= 9
10	$1x10^1 + 0x10^0$	1010	$1x2^3 + 0x2^2 + 1x2^1 + 0x2^0$	= 10
11	$1x10^1 + 1x10^0$	1011	$1x2^3 + 0x2^2 + 1x2^1 + 1x2^0$	= 11
12	$1x10^1 + 2x10^0$	1100	$1x2^3 + 1x2^2 + 0x2^1 + 0x2^0$	= 12
13	$1x10^1 + 3x10^0$	1101	$1x2^3 + 1x2^2 + 0x2^1 + 1x2^0$	= 13
14	$1x10^1 + 4x10^0$	1110	$1x2^3 + 1x2^2 + 1x2^1 + 0x2^0$	= 14
15	$1x10^1 + 5x10^0$	1111	$1x2^3 + 1x2^2 + 1x2^1 + 1x2^0$	= 15
16	$1x10^1 + 6x10^0$	10000	$1x2^4 + 0x2^3 + 0x2^2 + 0x2^1 + 0x2^0$	= 16
17	$1x10^1 + 7x10^0$	10001	$1x2^4 + 0x2^3 + 0x2^2 + 0x2^1 + 1x2^0$	= 17
20	$2x10^1 + 0x10^0$	10100	$1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 0x2^0$	= 20

Decimal numbers since they have a base of 10, may be broken up into powers of 10.

e.g. $305.798 = 3x10^2 + 0x10^1 + 5x10^0 + 7x10^{-1} + 9x10^{-2} + 8x10^{-3}$

In the same way, binary numbers, since they have a base of 2, may be broken up into powers of 2.

e.g. $101.011 = 1x2^2 + 0x2^1 + 1x2^0 + 0x2^{-1} + 1x2^{-2} + 1x2^{-3}$

It can be seen from this arrangement of the powers of the bases, that the decimal places (units, tens, hundreds, tenths, hundredths, thousandths, etc) have a definite relation to the powers of the base 10 in the decimal system. They are numbered off consecutively from left to right, from $+\infty$ to $-\infty$, these numbers corresponding exactly with the powers of the base 10; the decimal point is placed between the units (0) place and the tenths (-1) place.

The binary places (units, twos, fours, eights, sixteens, halves, fourths, eighths, etc.) are also numbered exactly according to the powers of the base 2, the binary point is placed between the units (0) place and the halves (-1) place. Therefore, the place and point arrangement is the same in both decimal and binary systems.

e.g.

Decimal Place and Binary Place No / No.	$+\infty$ . . . 3, 2, 1, 0, -1, -2, -3, ... $-\infty$
305 798	3 0 5, 7 9 8
101 011	1 0 1, 0 1 1

Conversion of Decimal Numbers to Binary Numbers - In order to convert a number from the decimal system to the binary system, the number must be changed from powers of 10 to powers of 2; therefore, powers of 2 are taken out of the decimal number. Follow the work sheet at the end of the three examples.

Example 1 Given: 18

To Find: Binary Equivalent

Method: Take out of the decimal number, 18, the highest power of 2, $= 16 = 2^4$. A 1 is now placed in binary place No. 4, corresponding to the power of 2 found. Subtracting 16 from 18, leaves 2. The highest power of 2 in 2 is $2^1 = 2$; therefore, put a 1 in binary place No. 1. Subtracting 2 from 2, leaves 0, so the conversion is completed. Since 4 and 1 were the only powers of 2 in the given number, 1's appear only in binary places 4 and 1. The coefficients of the non-appearing powers must have been zero, so zeros are entered under all the other binary place numbers.

Example 2

Given. 730

To Find. Binary Equivalent

Method. The highest power of 2 in 730 is $2^9 = 512$, so a 1 goes in No. 9
730 - 512 = 218

The highest power of 2 in 218 is $2^7 = 128$ " " " " " No. 7
218 - 128 = 90

The highest power of 2 in 90 is $2^6 = 64$, " " " " " No 6
90 - 64 = 26

The highest power of 2 in 26 is $2^4 = 16$, " " " " " No. 4
26 - 16 = 10

The highest power of 2 in 10 is $2^3 = 8$, " " " " " No 3
10 - 8 = 2

The highest power of 2 in 2 is $2^1 = 2$, " " " " " No 1

No other powers of 2 appear so their coefficients must be zero.

Example 3

Given. 10.625

To Find. Binary Equivalent

Method. The highest power of 2 in 10 625 is $2^3 = 8$, so a 1 goes in No 3
10.625 - 8 = 2.625

The highest power of 2 in 2.625 is $2^1 = 2$, " " " " " No 1
2.625 - 2 = .625

The highest power of 2 in .625 is $2^{-1} = .5$, " " " " " No (-1)
.625 - .5 = .125

The highest power of 2 in .125 is $2^{-3} = .125$, " " " " " No (-3)
.125 - .125 = 0.

which indicates completion of conversion. No other powers of 2 appear in 10 625, so their coefficients are zero.

WORK SHEET

Decimal Place No / No	10	9	8	7	6	5	4	3	2	1	0	-1	-2	-3	-4	-5	-6
.133	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	0
.730	0	1	0	1	1	0	1	1	0	1	0	0	0	0	0	0	0
0.635	0	0	0	0	0	0	0	1	0	1	0	1	0	1	0	0	0

Given 147

To Find Binary Equivalent

Method: Follow the table given below

The highest power of 2 in .142 is 2^{-3} = .125 so a 1 goes in No -3
147 - .125 = 022

The next power of 2 in order, is 2^{-4} = .0625, but this power of 2 is not contained in 022, so coefficient of the (4) place = 0

2^{-5} = .03125, not in 022, so 0 in No (-5)

2^{-6} = .015625, is in 022, so 1 in No. (-6)
(.022 - .015625 = .006375)

2^{-7} = 0078125, not in .006375, so 0 in No (-7)

2^{-8} = 00390625, in .006375 so 1 in No (-8)
(.006375 - .00390625 = .00246875)

2^{-9} = 0019531.25, in 00246875 so 1 in No. (-9)
(00246875 - .0019531.25 = .000515625)

2^{-10} = .0009765625, not in 000515625 so 0 in No (-10)

2^{-11} = .00048828125 in 00515625 so 1 in No (-11)
(000515625 - 00048828125 = 00002734375)

2^{-12} = .000244140625, not in 00002734375, so 0 in No. (-12), etc

Place / No	0	-1	-2	-3	-4	-5	-6	-7	-8	-9	-10	-11	-12
147	0	0	0	1	0	0	1	0	1	1	0	1	0

That method of converting a decimal to the binary system always works, but it is laborious and offers many chances for mistakes in the division and subtraction of such long numbers. There is another method based on the same principle of taking out powers of 2, which, however, is much simpler. Given a decimal: by the former method, if it is larger than 2^{-1} (= .5), a 1 goes in the -1 place; if the number (or the remainder after subtraction of .5) is greater than or equal to .25 (= 2^{-2}), a 1 goes in the -2 place. However, it is the same thing to say, if twice the given decimal is larger than 2×2^{-1} (=1), a 1 is put in the -1 place; if 4 times the given decimal is larger than 4×2^{-2} (= 1), a 1 is put in the -2 place. If 8 times the given decimal is larger than 8×2^{-3} (= 1), a 1 is put in the -3 place. This is the same thing as doubling the number (or its remainder after a power of 2 is taken out) at each step and comparing it with 1. If the result becomes greater than 1, a 1 is taken out, and the doubling process starts again on the remainder.

Given: .147

To Find: Binary Equivalent

Method: The former method started out by asking:

Is $.147 \geq .5$? (If so, a 1 goes in No. -1; if not, a 0 goes in -1.

Is $.147 \geq .25$? (" " " " " No. -2; " " " " " -2.

Is $.147 \geq .125$? (" " " " " No. -3, " " " " " -3

This method starts out by asking:

Is $2(.147) \geq 1$? (If so a 1 goes in No. -1; if not, a 0 goes in -1.

Is $2 \times 2(.147) \geq 1$? (" " " " " -2; " " " " " -2.

Is $2 \times 2 \times 2(.147) \geq 1$? (" " " " " -3; if not, " " " " -3.

$2(.147) = .294 \not\geq 1$, therefore, 0 in No. -1
$2 \times 2(.147) = .588 \not\geq 1$, " " " No. -2
$2 \times 2 \times 2(.147) = 1.176 \geq 1$, " " " No. -3

$(1.176 - 1.000 = .176)$

$2(.176) = .352 \not\geq 1$, therefore, 0 in No. -4
$2 \times 2(.176) = .704 \not\geq 1$, " " " No. -5
$2 \times 2 \times 2(.176) = 1.408 \geq 1$, " 1 " No. -6

$(1.408 - 1.000 = .408)$

2(608) = 816 ≯ 1, therefore, 0 in No. 7
2 x 2(608) = 1 632 ≯ 1, " 1 in No. 8

(1 632 - 1 000 = 632)

2(632) = 1 264 ≯ 1, therefore, 1 in No. 9

(1 264 - 1 000 = 264)

2(264) = 528 ≯ 1, therefore, 0 in No. 10
2 x 2(264) = 1 056 ≯ 1, " 1 in No. 11

(1 056 - 1 000 = 056)

2(056) = 112 ≯ 1, therefore, 0 in No. 12

etc.

This result checks with the previous one. It can also be shown that if a decimal repeats itself in the decimal system, it also repeats itself in the binary system.

Conversion of Binary Numbers to the Decimal System - Given a number in the binary system, it is always a simple matter to convert it to the decimal system. The converted number is simply the sum of the powers of 2 whose presence in the given number is indicated by 1's in the corresponding binary places.

Binary Place	5	4	3	2	1	0	-1	-2	-3	-4	Decimal Equivalent
	1	0	1	1	0	1	0	1	0	1	$1x2^5 + 1x2^3 + 1x2^2 + 1x2^0 + 1x2^{-2} + 1x2^{-4} =$
											32 + 8 + 4 + 1 + .25 + .0625
											45.3125

The coefficients of the other powers of 2 are zero, so they do not contribute to the converted number.

Addition - Since 1 is the largest digit in the binary system, it is evident that any sum larger than 1 must be represented with the aid of carryovers. Therefore, no matter how many 1's are added up in one column, the result under that column must be a 0 or a 1; the rest of the sum is carried over in its binary notation and set up at the head of the adjacent columns to the left, as carryover figures. Thus, if a sum of 1's in a column adds up to 6 (which is 110 in the binary notation) a 0 is put at the bottom of the column and the two 1's are put at the head of adjacent columns to the left as carryovers. This is the same as adding the 1's in binary fashion at each step of the columnar addition.

(1 + 1 = 10; 10 + 1 = 11; 11 + 1 = 100; 100 + 1 = 101; 101 + 1 = 110 = 6)

e.g.	1	1	10	2	11	3	111	7	11101	29	
	1	1	1	1	1	0	101	5	11011	27	
	10	2	11	3	100	4	1100	12	110010	56	

(The small numbers above the examples are carry-over figures put in for sure in following the addition procedure.)

							or, more easily				
e.g.	0 111	7	0 111	7			111	7	111	7	Addends
	011	3	110	6			011	3	110	6	
	101	5	011	3			1010	10	1101	13	
	001	1	111	7			101	5	011	3	
	100	4	010	2			1111	15	10000	16	
	111	7	100	4			001	1	111	7	
	11012	27	11101	29			10000	16	10111	23	
							100	4	010	2	
							10100	20	11001	25	
							111	7	100	4	
							11011	27	11101	29	Sum

Subtraction - Subtraction is based on the following rules:

A 0 from 1, always gives 1, and a 1 from 0, always gives 1, but the latter requires "borrowing" from the first column to the left. A 1 in the first column to the left is reduced to 0 by borrowing, a 0 in the first column to the left is reduced to 1, causing the digit in the second column over to be reduced, etc.

	0		0		0		0		0 0 0		
	10	2	100	4	1000	8	1110	14	1001010	74	Minuend
	1	1	1	1	1	1	1	1	0111101	61	Subtrahend
	1	1	011	3	0111	7	1101	13	0001101	13	Remainder

The small numbers show how the digits are changed by borrowing. See also Subtraction under "Complements"

Multiplication - Multiplication in the binary system is done exactly as in the decimal system and is based on the multiplication table 0 x 1 = 0, 1 x 1 = 1, 0 x 0 = 0.

101011	or	101011	$43 = 2^5 + 2^3 + 2^1 + 2^0$	Multiplicand
101110		101110	$46 = 2^5 + 2^3 + 2^2 + 2^1$	Multiplier
000000		101011	258	
101011		101011	172	
101011		100000010	1978	Product
101011		101011		
000000		1001011010		
101011		1010110	$1978 = 2^{10} + 2^9 + 2^8 + 2^7 + 2^5 + 2^4 + 2^3 + 2^1$	
11110111010		11110111010		

1) Division - Division in the binary system is carried out exactly as in the decimal system.

1) $1^{1}\overline{1} = 1$ $1^{1}\overline{0} = 0$

```
                 0111 1100100000011001
          10111)10110011 0000000000000000
                 10111
                 0101011
                  10111
                  0101001
                   10111
                   0100100
                    10111
                    0011010
                     10111
                     00011000
                      10111
                      0000100000
                          10111
                          00100100
                           10111
                           0011010
                            10111
                               11000
                               10111
                                0001
```

```
7.783608
23)179.000000
   161
   180
   161
   190
   184
    60
    46
   140
   138
   200
   184
    16
```

Check on Quotient

$2^{3} - 2^{1} + 2^{0} = 7$

2^{-1}	= .5
2^{-2}	= .25
2^{-5}	= .031250
2^{-10}	= .000976
2^{-12}	= .000244
2^{-13}	= .000122
2^{-16}	= .000015
	7.783607

It should be noted that in order to get the decimal equivalent of the binary quotient to equal the decimal quotient to 5 decimal places, the binary division had to be carried to 16 binary places.

Complements - The ordinary complement of a number in the decimal system is obtained by subtracting the number from the next higher power of 10 e.g. Complement 18 = 100 - 18 = 82. The ordinary complement of a number inthe binary system is obtained by subtracting the number from the next higher power of 2. e.g. Complement of $5 = 2^{3} - 5 = 8 - 5 = 3$ It can be shown that the ordinary complement of a power of 2 is that power of 2 itself. See Example 3.

Another kind of complement of a number is obtained by subtracting the number from any higher power of 2. Notice its used under "Complements" (a. Subtraction).

	(1)	(2)	(3)
Given	100101	101010	100000
To Find	Binary Complements		
Method	Subtract from next higher power of 2		

(1)	1000000	64	(2)	1000000	64	(3)	1000000	64
	-100101	-37		-101010	-42		-100000	-32
	0011011	27		0010110	22		0100000	32

Another method for finding the ordinary complement of a number in the binary system is to interchange all 0's and 1's and add 1

	(1)	(2)	(3)
Given:	100101.	101010	100000

To Find: Binary Complements

Method: Interchange 0's and 1's and add 1.

(1) No. 100101 number
011010 interchange 0's and 1's
1 add 1
011011 complement

(2) No. 101010
010101 "
1
010110

(3) No. 100000
011111 "
1
100000

These results check with those above.

(a) Subtraction

Instead of subtracting one number from another it is possible to take a complement of the subtrahend and add that complement to the minuend, provided the power of 2 which was added to the subtrahend in order to get a complement is subtracted from the answer. Practically, subtracting out the added power of 2 means dropping the 1 in the last binary place on the left, if the power of 2 used in getting the complement is greater than that contained in either number. If not, then the power of 2 must be subtracted out by the usual subtraction method.

e.g. 1011100 - 10011 = 1001001
subtrahend

{1011100 + (100000 - 10011) = 1101001 = 1001001 + 100000
a complement of subtrahend

{1011100 + (10000000 - 10011) = 11001001 = 1001001 + 10000000

{1011100 + (10000000000 - 10011) = 10001001001 = 1001001 + 10000000000}

Regular Subtraction

1011100	
- 10011	Number
1001001	Answer

Subtraction by Addition of Complements

1011100		1011100	1011100
01101	Complement	1101101	1111101101
1101001	Sum	11001001	10001001001
-100000		10000000	10000000000
1001001	Answer	01001001	00001001001

Notice that in the two examples on the right, dropping the last 1 on the left in the sum given the same result as subtracting out the power of 2 added to get a complement, because the power of 2 added was greater than that contained in either number.

Raising Numbers to Powers

$111^{10.1} = 7^{2.5}$

$\log_{10} x = \log_{10} 7^{2.5}$

$x = \log_{10}^{-1} (2.5 \log_{10} 7)$

Binary numbers probably would have to be converted to the decimal system for intelligibility in working with their non-integral powers.

Logarithms to the Base 2

$\log_2 1$	=	0	$\log_2 8$	=	3
$\log_2 2$	=	1	$\log_2 16$	=	4
$\log_2 4$	=	2	$\log_2 32$	=	5 etc.

$\log_2 15 = ? = 3^+$

$\log_2 N = x \qquad N = 2^x$

$\log_{10} N = y \qquad N = 10^y$

Therefore $2^x = 10^y$

$\log_{10} 2^x = \log_{10} 10^y$

$x \log_{10} 2 = y \log_{10} 10$

$x \log_{10} 2 = y$

$(\log_2 N)(\log_{10} 2) = \log_{10} N$

$$\log_2 N = \frac{\log_{10} N}{\log_{10} 2}$$

Square Roots

Square roots may be obtained in the binary system in exactly the same manner as in the decimal system

e.g.

```
          1  1  1
        √11,00,01          7.0
          01              √49
    101   1000            49
          101        140   0
   1101   001.101
            1101
               0
```

```
          1  1  0  0          1 2
        √10,01,00,00        √1,44
          1                   1
    101   0101          22    0 44
          101                 44
             0                 0
```

```
             0 1                2 5
           √.0001             √.0625
             00                 4
      0      001          45    225
             001                225
                                  0
```

In the binary system, there is also another way of taking square roots. Follow the example outlined below:

1) The radicand is marked off in pairs to the left and right of the binary point

2) Twice the root so far obtained, with a 1 placed under the right hand digit of the first non-zero pair in the radicand, (2r, 1), is subtracted from the first pair of digits where both digits $\neq 0$.

3) If the remainder is ≥ 0, a 1 is placed in the root place above the pair. If the remainder is < 0, a 0 is placed in the root, the previous 2r, 1 is removed, and the next pair of digits is brought down from the radicand. As before, twice that total root with a 1 in the last place on the right, (2r, 1), is subtracted from the previous remainder with digits brought down. In other words, 2r, 1 is subtracted at each step in this rooting process where r is the total root so far found.

Given: $\sqrt{11001}$

To Find: The square root by the second method

Method: (1) $\sqrt{1,10,01}$ (1) The digits in the radicand are marked off in pairs to the left and right of the binary point.

(2) $\sqrt{11001}$
$\underline{1}$
0 (2) (2r,1) is subtracted from the first pair of (non-zero) digits. (2r,1) = 0, 1 (The root so far is zero because no root has been found as yet!)

(3)
$$\begin{array}{r} 10 \\ \sqrt{11001} \\ \underline{1} \\ 010 \\ \underline{101} \\ 11111111101 \end{array}$$

(3) The remainder is zero, so a 1 goes in the corresponding root place above the first "pair". The next 2 digits are brought down. 2r, 1 is subtracted from the remainder with digits brought down.
2r, 1 = 2(1), 1 = 10, $\bar{1}$
The remainder this time is negative, therefore, a zero goes in the second root place and the previous 2r, 1 is removed.

```
      1 0 1
(4)  ∛1001
     1
     0100 1
      100 1
        0 0 0
```

(4) The next 3 digits are brought down. 3r, 1, is subtracted from the remainder with digits brought down.
3r, 1 = 3(10), 1 = 100, 1
This remainder is zero, so a 1 goes in the third root place. The remainder is zero and there are no more pairs of digits to be brought down so the rooting process is complete.

Engineer Margaret Thornwood

Approved

MIF:hes
February 18, 1946

6345
Report No. R-115

SERVOMECHANISMS LABORATORY
Massachusetts Institute of Technology
Cambridge, Massachusetts

Date of Report: February 17, 1947 Page 1 of 6 pages

Subject: Electronic Digital Computer Development at M.I.T.

References: The following text was presented as a conference paper before the American Institute of Electrical Engineers at their Winter Meeting, Thursday, January 30, 1947 in New York City.

Written by: Jay W. Forrester

Since the time that Dr. Vannever Bush designed the first differential analyzer at the Massachusetts Institute of Technology, research there has been active on computing aids to the engineer and scientist. The present development of a high-speed electronic digital computer results naturally from the pre-war research of the Electrical Engineering Department and from the wartime developments at the M.I.T. Servomechanisms and Radiation Laboratories. The program at the M.I.T. Servomechanisms Laboratory to develop a high-speed electronic digital computer is sponsored by the Special Devices Division of the Office of Naval Research.

The fields of pure and applied science and engineering today face many problems, the solution of which will be practical only when new and tremendously more powerful automatic computing machines are available. Several digital computers have already been built at various laboratories in the United States. These are already known or are discussed in other papers at this convention. Existing techniques must be extended toward higher computing speeds and greater internal storage capacity to solve problems of ever increasing urgency. Amongst these problems are subsonic and supersonic airflow around highspeed aircraft; calculations of stress and vibrational characteristics of structures; and solution of electric circuits, servomechanisms and automatic control systems. Outside of engineering are important applications to statistical analysis as encountered by the Department of Agriculture, Public Health, and the Bureau of the Census.

Application of electronic digital computers will in time be made to control of chemical plant processes and to what is often called the "automatic factory".

Before discussing technical highlights of the research at M.I.T., let us recall three important characteristics of a computer

of the type being considered.

1. First, the setup of a new problem is accomplished without physical change to the computer. That is, mechanical settings or switches are not used to describe the problem and control the solution. The control program for a new problem is supplied as data on magnetic tape or photographic film. This film is then read into the internal memory of the computer where any order can be obtained within a matter of microseconds under automatic control of the computer itself.

2. Second, the computer carries out a series of arithmetical operations one at a time. It is therefore equivalent, except in speed to a human operator who does one operation at a time while writing each partial result in a notebook.

3. Third, such a computer, besides the basic arithmetical operations of addition and multiplication, will be able to make choices. The element of alternate choice is a nearly indispensible feature. This choice permits the computer, as often as desired, to select one of two alternate computing sequences depending on the outcome of the computed results in the problem itself. A companion feature to making alternate choices is the ability to insert computed results into the controlling program. For example, choice of alternate computing programs arises in computation involving termination of an iterative procedure after the required convergence of the numerical results has been achieved. A simpler and more obvious application arises in engineering problems involving the discontinuities of back lash and Coulomb or static friction.

   Insertion of computed results into the control program of the computation is necessary, for example, in interpolation. The interpolation process requires use of computed results as a control code to locate associated values of an arbitrary function. Selected values of the function are then inserted into the desired interpolation formula.

These three features result in a flexibility not known in existing computers.

Let us now turn our attention to the research and development program at M.I.T.

For proper perspective, it should be stated that the electronic computer research which is described is in its intermediate stages and that a completed system does not yet exist. In fact the entire field of electronic digital computers has only been opened in a practical way and several years will be required to explore many of the engineering applications to which I will refer. The size or rating of a digital computer can be given by two values: the internal number storage capacity, and the computing speed. Storage capacity is a measure of the computer's ability to retain control program and partial numerical results for future use. It corresponds to the notebook record kept by the human operator of a desk calculator. The automatic computer prints out final results onto paper or film and discards partial results no longer needed. Storage must hold only the data actually required for future computation.

The computer being designed at M.I.T. is expected to have a storage capacity of 16,000 numbers of 40 binary digits each. This corresponds to 16,000 numbers of 12 decimal places each.

Certain objectives of the work will require computing speeds which are high even for electronic computers. Multiplication of two numbers of 12 decimal places each should require less than 50 microseconds. This means that on the average some 20,000 to 40,000 arithmetical operations can be performed per second.

It is obvious that such computing speeds are essential only where a long sequence of computations must be repeated tens or hundreds of thousands of times. Situations of this kind arise in the solution of partial differential equations and certain engineering applications where the computer will control mechanical or electrical devices operating in real time.

The arithmetic element of the computer will carry out the basic operations of addition and multiplication and probably also division. It will make choices of program and accomplish substitution orders as already described. All other operations such as integration, differentiation, and taking roots of numbers can be readily accomplished by iterative or series procedures using these basic arithmetic operations.

In designing a computer a wide choice in block diagrams is available. The computer herein discussed will use the binary system of numbers based on powers of 2 rather than powers of 10. Since only the digits 0 and 1 appear, the binary system lends itself well to electronic circuits involving trigger and gate tubes. Usefulness of the base 2 can be appreciated by recalling that the multiplication table is reduced to 1 times 1 equals 1 and all other products equal zero. In the computer,

digits of a number will be transmitted simultaneously over parallel buses from one part of the computer to another.

Final design of an electronic computer for the characteristics mentioned will be possible only after the perfection of many new techniques in video circuits and data storage devices. It is anticipated that some three years still will be required for the research, design, and construction of this device.

Methods are being tested for generation, control, and gating of video pulses of 0.1 microsecond duration at a 5 megacycle repetition rate. Improvements must be made in pulse transformers, switching, and arithmetical circuits. To permit step-by-step manual control of the computer for maintenance purposes most circuits must have d-c coupling. At the same time the circuits must possess a band width suitable for 0.1 microsecond video pulses. Pulses appear in noncyclic manner and most circuits must be independent of repetition rate.

A suitable number storage system is being developed. It is obvious that storage of 600,000 binary digits is not practical with individual vacuum tubes. A bulk storage method is required. The digit storage method for the M.I.T. computer will be in electrostatic form. The digits 0 or 1 will be stored as positive and negative points of charge on a dielectric insulating plate. These charges can be placed on a dielectric surface with a cathode ray beam. At a later time, the beam can return to the same location for selecting and reading the stored signal into an external circuit. One can expect to store charges approximately 1/8 inch apart on a dielectric surface with a suitable tube. Storage of signals for sufficient periods of time has already been demonstrated and clear output signals can be obtained. Charge is removed from the dielectric by secondary emission as mentioned yesterday morning in the paper by Dr. Du Bridge. As yet, several problems in the collection and control of these secondary electrons have not been solved.

Reliability is a matter of major concern in computer research. In many electronic and communication problems, static, noise, or other momentary disturbances can be tolerated. Here, however, we find that the momentary appearance of an extraneous signal may alter the order of magnitude of a numerical value. Wide tolerance ratios must be provided between signal and noise level and between transmitter and receiver signal specifications. Checking circuits must be built into the equipment wherever possible to detect and indicate improper operation. In spite of the severe video band width requirements, vacuum tubes must, wherever possible, be operated at less than rating.

Let us now turn our attention to applications of the high speed electronic digital computer. These can be divided broadly

between mathematical and control computation. By mathematical computation is meant the solution of individual problems pertaining to the fields of mathematics, physics, and engineering and is the application ordinarily visualized.

Control computation can be the repeated solution of the same problem for directing the operation of a physical system as for example the control of a chemical process.

In pure mathematics high-speed computers will permit exploratory solutions, on a numerical basis, of non-linear and partial differential equations. These specific results may then lead to a sufficient understanding of the original equations to permit formulation of explicit mathematical solutions as has long ago been done for the simpler linear equations.

In engineering problems the digital computer which is easy to set up and has high computing speeds is well adapted to solution of non-linear and discontinuous systems.

Mr. Dunstan in his paper on "Machine Computation of Power Network Performance" has discussed numerical solution of a-c power systems.

Mr. Dunstan commented that problem planning and setup required a large percentage of the total time consumed.

With the new electronic computers, we can anticipate the reduction of setup time through the use of generalized problem programs each of which would fit a large class of engineering situations. Such generalized programs to be kept in a library file might be inefficient from the viewpoint of a specific problem. However, the high-speed computer greatly reduces the emphasis on computing efficiency and shifts the emphasis to ease of setup. As specific examples one could have available programs for the solution of an nth order system of algebraic equations with real or complex coefficients. Solution of such a system of equations would require only entry of the coefficients and the numerical value of n. Likewise, the multiplication of matrices could be so programmed that only the entry of numerical values and the dimensions of the matrices would be required.

After digital computers have been demonstrated in the field of mathematical computation will come their application to engineering control. Following the development of the large and rather complex units which we have discussed, there can be forseen a period of

simplification and re-design for cost reduction which will make the computer practical for many forms of process control with application to manufacturing, chemical plants, and power generating stations.

Jay W. Forrester

Jay W. Forrester

JWF:mla

6341
Report No. R-115

SERVOMECHANISMS LABORATORY
Massachusetts Institute of Technology
Cambridge, Massachusetts

Date of Report: March 3, 1947 — Page 1 of 5 pages

Subject: Electronic Digital Computers — Drawings: A-30337, A-30338, A-30339

Reference: The following text was prepared for the Institute of Radio Engineers at their Winter Meeting, Tuesday, March 4, 1947 in New York City.

Written by: Jay W. Forrester

This paper is to be an introduction for today's session on electronic digital computers rather than a discussion of the high-speed computer research program at M.I.T.

Before embarking on the subject of digital computers, we should first clearly separate the two broad classes of computers -- the digital machines as distinguished from analog machines. Analog computers measure physical quantities such as mechanical motions or electrical voltages to represent numerical values of the computation. Examples are to be noted in the mechanical or electrical fire control computers, the differential analyzer and the electrical a-c network analyzer. In the analog machine a computing element such as an adder or an integrator is required for each mathematical operation in the problem solution. Such a collection of computing elements is indicated in Drawing A-30337. In the analog computer, the problem to be solved is described by the physical connection of the elements.

A network of connections describing a particular problem is shown in Drawing A-30338. The analog computer when set up for an ordinary differential equation gives theoretically a correct solution. In practice, however, the solution is limited in accuracy by the physical tolerances of the component mechanisms and by stray signals, mechanical back lash, and noise. Precision of one part in one thousand is good and perhaps one part in ten thousand is the maximum obtainable.

On the other hand a digital computer uses numbers on a numerical or arithmetical basis. Numerical values are handled as digits rather than physical measurements and, as in the desk calculating machine, one can provide for as many decimal places as desired. The solution of an ordinary differential equation by numerical methods is theoretically inexact because a continuous process has been replaced with a step-by-step process. The numerical solution, however, can be made arbitrarily as good as desired by selection of more decimal places in the calculations

and a shorter interval between solutions.

In Drawing A-30339 is the block diagram of a digital computer of the type being described at the convention today. Physical interconnection of the computer is fixed and independent of the problem being solved. Each element can transmit or receive numbers on a central bus system. Operation is mathematically identical to a human operator with a desk calculating machine and notebook.

The plan or sequence of calculation is known as the control program and is stored as a series of coded signals to represent orders for arithmetical operations. The control element receives from storage this coded information describing the series of numerical operations to be performed. In response to these signals, the control transfers numbers in digital form between the remaining elements of the computer.

The arithmetic element of the computer is limited to a few basic functions, such as addition, subtraction, multiplication, and division. All other operations, as for example integration and extraction of roots, are accomplished by series or iteration methods. The arithmetic element performs a single operation at a time, receiving inputs from storage and sending results to storage.

Two other nearly indispensible operations are performed by the arithmetic element. One is the exercise of choice for the selection of alternate computing sequences depending on numerical results in the computation. The other essential operation is the substitution order permitting the insertion of a computed result into the controlling program. For example, choice of alternate computing programs arises in computation involving termination of an iterative procedure after the required convergence of the numerical results has been achieved. A simpler and more obvious application arises in engineering problems involving the discontinuities of back lash and Coulomb or static friction. The substitution order can be illustrated in the process of interpolation. A numerical value of the argument, for example x, arises from the computation process. This value of x must be used in the control program to locate the nearest value of a function of x,

$$\left[f(x_1),\ f(x_2),\ f(x_3) \right]$$

located in storage. The selected values of $f(x_n)$ are inserted into any desired interpolation formula for calculation of $f(x)$.

Storage, which in electronic computers will probably be of the electrostatic form, is perhaps the most important part of the computer. In storage is retained the control code for the computation as well as partial results. It corresponds to the notebook kept by the human computer. One form of electrostatic storage will be described in the paper by Dr. Rajchman.

The input mechanism provides communication between the human operator and the computing machine. It may be in the form of magnetic tape, punched tape, or photographic film. The control code and initial data of the problem prepared by an operator are transferred to the storage from the input reading mechanism. Mr. Alexander will discuss input mechanisms in more detail.

Output from the computer can be in several forms. For human use the output can be presented graphically by photographic means or in the form of printed numbers. For future use by the computer itself, the output can be in coded numerical form on magnetic tape or photographic film.

Numbers are transmitted as video pulses coded in groups. In some computer systems, digits are transmitted in serial fashion following one another on the same signal line. In other systems, digits in all columns are transmitted simultaneously over parallel conductors.

Most research groups favor the binary system of number notation in which only the digits zero or one are used. In the binary system the two digits can be represented by either the presence or absence of a video pulse at the proper instant of time. Vacuum tube circuits need exist in only two states and the freedom from graduated signals greatly enhances reliability.

Historically, the first attempt to build a digital computer was financed by the British government in about 1835. The construction of a mechanical computer, controlled by a sequence of punched cards failed at that time because of inadequate machine tools and production methods. The mechanical computer progressed over the last hundred years from the adding machine, through the desk calculator and the punched card business machine, to the Automatic Sequence Controlled Calculator at Harvard University.

The mechanical calculator is, however, much too slow operation and limited in capacity for many of the problems which today face science and engineering. As engineering activities become more complex and as scientific and mathematical studies become more daring, we encounter a need for greater and greater computer speed, capacity, and flexibility. Mr. Crawford will discuss for you typical applications of digital computers requiring high computing speed and storage capacity.

The first departure from the mechanical computer is perhaps found in the Bell Telephone Laboratories relay computers which are tape controlled and the University of Pennsylvania Eniac which is electronic but is however, set up for a new problem by manual switch and plugboard.

From the wartime developments in electronics have come circuit elements and techniques to permit extension of computer ratings over those now existing by several orders of magnitude.

The size or rating of a digital computer can be described by two quantities: the internal storage capacity, and the computing speed. Storage capacity measures the computer's ability to retain controlling program information and partial numerical results. Computing speed measures the practicability of undertaking solutions requiring millions of arithmetical operations. While existing computers have an internal storage limited to a few thousand digits, electronic developments will permit storage of hundreds of thousands of digits. Where computing speeds were limited in mechanical and relay computers to a few arithmetical steps per second, electronic video circuits will push the limit to some 30,000 arithmetical operations per second.

It is apparent that computing speeds of many thousands of operations per second are essential to the solution of problems requiring hundreds of thousands or millions of operations. Situations of this kind arise in partial differential equations describing the flow of fluids and heat and the study of electromagnetic radiation. Lengthy series of calculations are likewise encountered in stress and deflection studies of structures and in the behavior and stability of complex automatic control systems.

Realization of computing speeds of thousands of operations per second will be accomplished through controlled video pulses of one to several megacycles repetition rate. Computer research is directed toward improvements in digit storage methods, pulse transformers, switching, and arithmetical circuits. In some circuits pulses of 0.1 microsecond duration at a 5 megacycle repetition rate will be required. To permit step-by-step manual control of a computer for maintenance purposes many of these circuits must be d-c coupled and yet have a band width to pass 0.1 microsecond pulses. Pulses in most circuits normally appear in noncyclic manner so that operation must be independent of repetition rate.

Radar and television research of the last few years has established the foundation for such systems. Extensive use will be made of pulse transformers, crystal diode rectifiers, and electrical delay lines.

Developments in vacuum tubes leave much to be desired in reliability and especially in the knowledge of those factors that influence and predict tube life. In the high-speed electronic computer there will be two to ten thousand vacuum tubes, the failure of any one being sufficient to stop the computer.

Reliability is a matter of major concern in computer research. In many electronic and communication problems, static, noise or

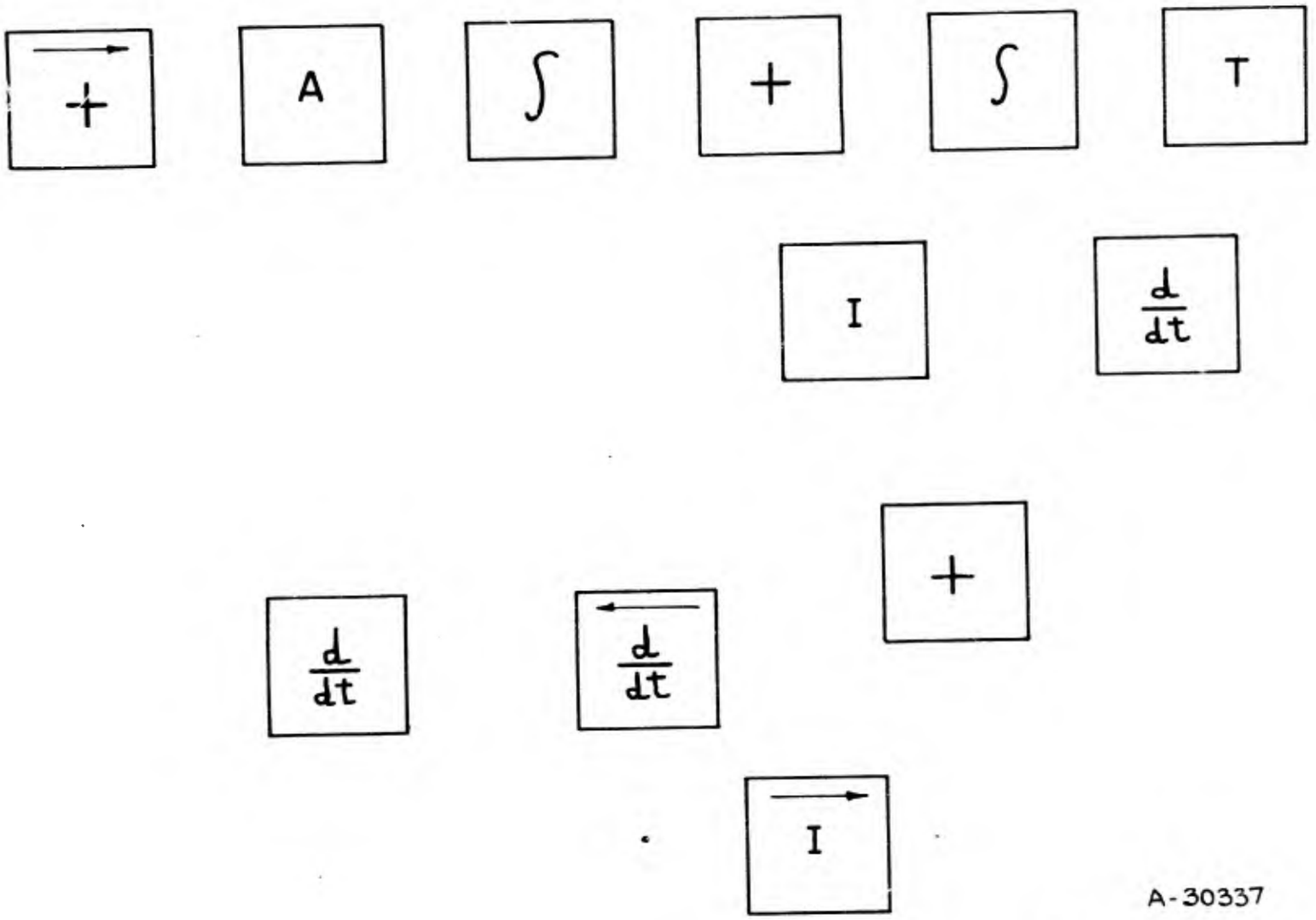
other momentary disturbances can be tolerated. Here, however, we find that the momentary appearance of an extraneous signal may alter the order of magnitude of a numerical value. Wide tolerance ratios must be provided between signal and noise level and between transmitter and receiver signal specifications. Checking circuits must be built into the equipment wherever possible to detect and indicate improper operation. In spite of video band width requirements of 10 to 20 megacycles, vacuum tubes must, wherever possible, be operated at less than rating.

The flexibility of a computer, such as herein discussed, distinguishes it from previously available equipment. The machine is controlled by data available to it from tape or film and physical arrangement of the computer is not altered when a new problem arises. Problems can be setup in rapid succession and the operator can return to any previous problem in a matter of seconds. A library of control films can be accumulated and selected to fit new problems.

With the new electronic computers, we can anticipate the reduction of setup time through the use of generalized problem programs each of which would fit a large class of engineering situations. Such generalized programs to be kept in a library file might be inefficient from the viewpoint of a specific problem. However, the high-speed computer greatly reduces the emphasis on computing efficiency and shifts the emphasis to ease of setup. As specific examples one could have available programs for the solution of a nth order system of algebraic equations with real or complex coefficients. Solution of such a system of equations would require only entry of the coefficients and the numerical value of n. Likewise, the multiplication of matrices could be so programmed that only the entry of numerical values and the dimensions of the matrices would be required.

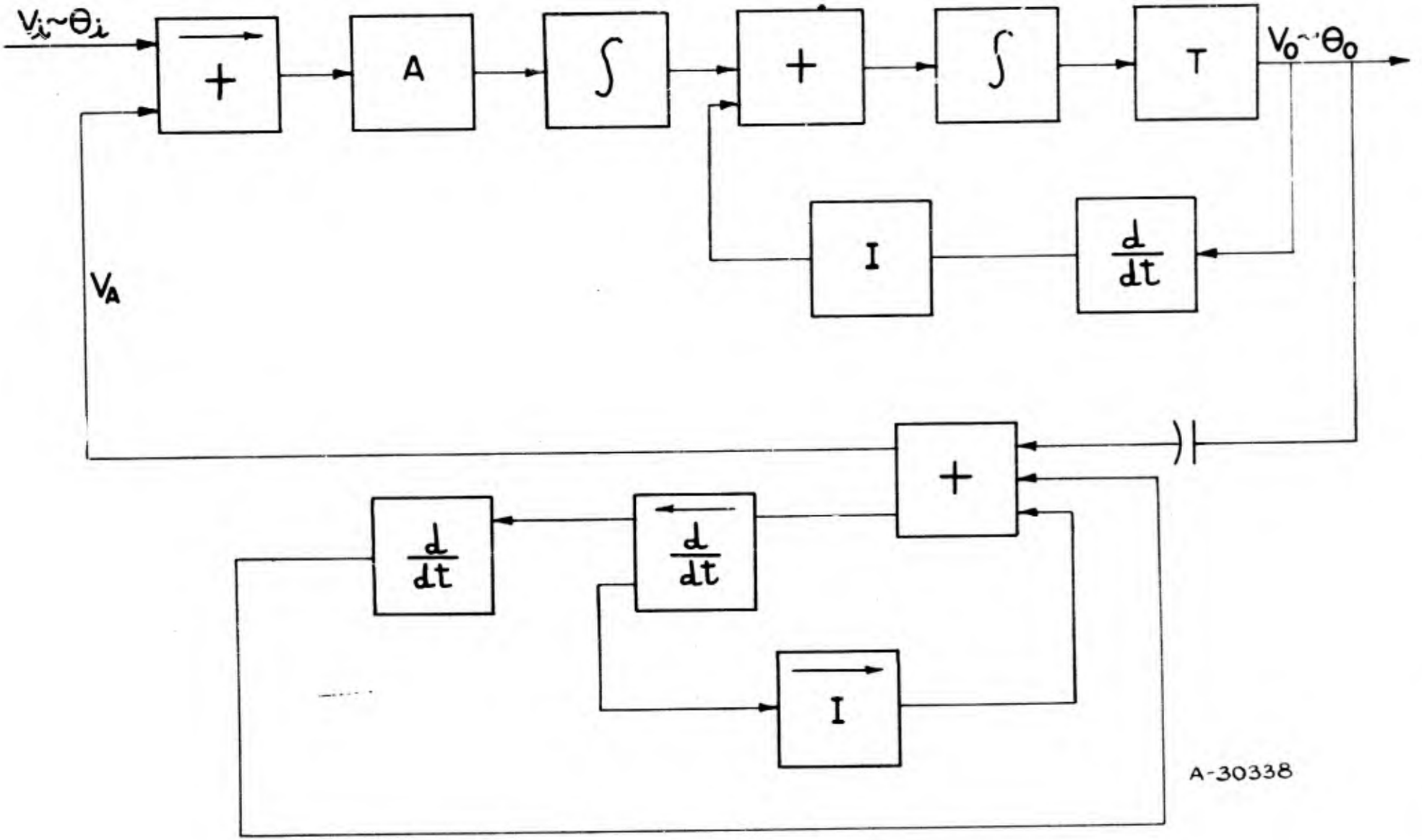
Jay W. Forrester

JWF:mla

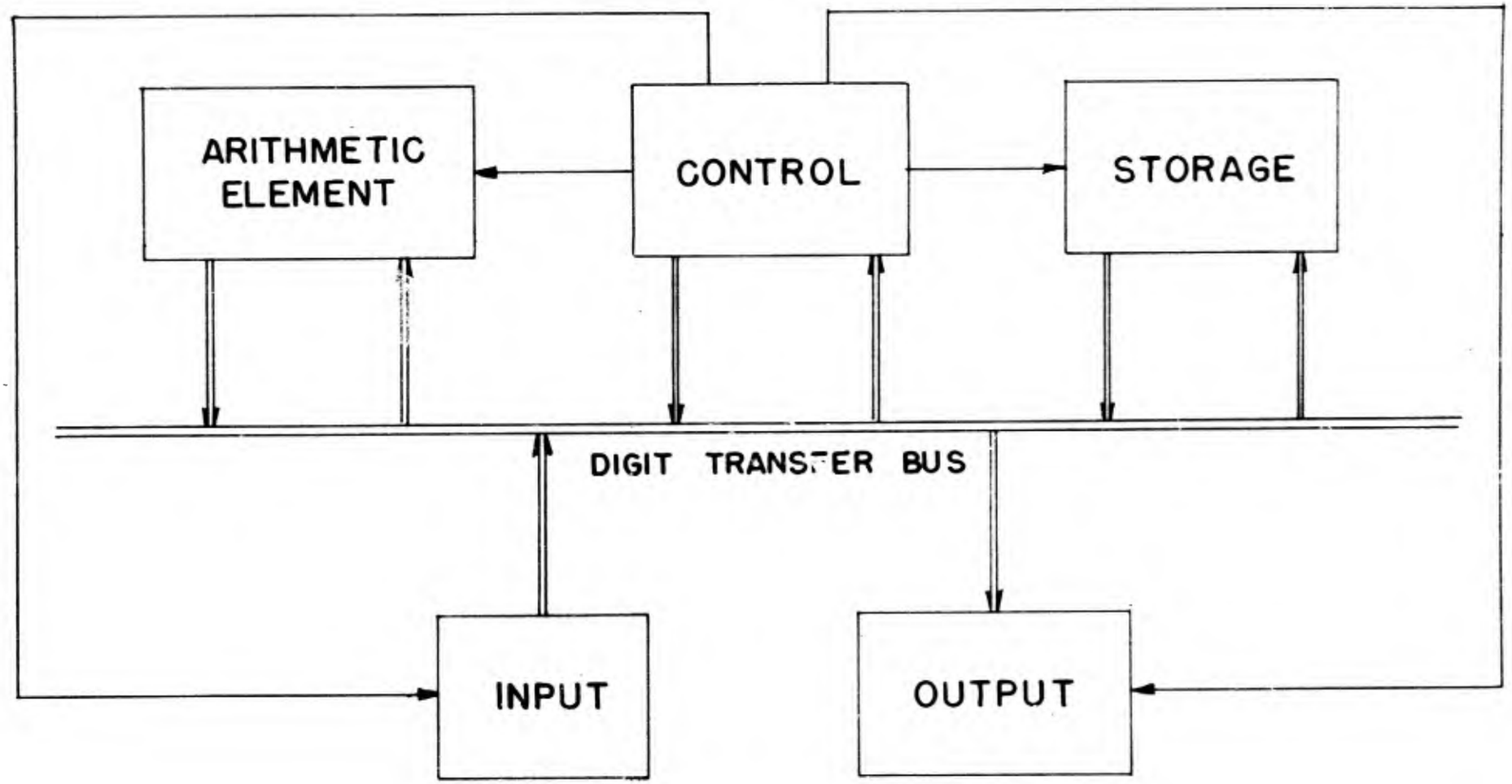


A-30337

6345 A 30337


$V_i \sim \theta_i$
+
A
∫
+
∫
T
$V_o \sim \theta_o$
I
$\frac{d}{dt}$
V_A
+
$\frac{d}{dt}$
$\frac{d}{dt}$
I
A-30338

A-30339-1



6345 RLK.
PRCE

A-30339-1